



IBM WebSphere Transcoding Publisher V1.1

Extending Web Applications to the Pervasive World

Enable universal data access to
improve business communications

Write once - reach anyone,
anytime, anywhere

Quickly support your
pervasive devices



Juan R. Rodriguez
Eduardo Azara
Lucia Ruiz
Pekka Kaukonummi

ibm.com/redbooks

Redbooks



International Technical Support Organization

IBM WebSphere

Transcoding Publisher V1.1

Extending Web Applications to the Pervasive World

July 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix D, "Special notices" on page 311.

First Edition (July 2000)

This edition applies to Version 1.1 of IBM WebSphere Transcoding Publisher, Program Number 5639-I48 for use with the Windows NT Operating System.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 678
P.O. Box 12195
Research Triangle Park, NC 27709-2195

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 2000. All rights reserved.

Note to U.S Government Users – Documentation related to restricted rights – Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Preface	ix
The team that wrote this redbook	ix
Comments welcome	xi
 Chapter 1. Introduction	1
1.1 Overview	1
1.2 Hardware and software prerequisites	2
1.3 Preference profiles	3
1.4 Transcoders	4
1.5 Running as a proxy	4
1.5.1 Stand-alone proxy	5
1.5.2 Caching proxy	5
1.5.3 Connecting to a firewall	6
1.6 Running as servlets (WAS filtering)	6
1.7 Running as JavaBeans	7
1.8 Administration Console	7
1.9 Toolkit	8
1.10 Integration with Host Publisher	9
1.11 Tracing and logging	9
1.12 National language support	10
1.13 WTP 1.1.1 CSD	11
1.14 Resources used in this redbook	11
 Chapter 2. Architecture	13
2.1 Overview	13
2.2 Transcoding structure	14
2.2.1 Pluggable framework	15
2.2.2 IBM Transcoder plug-ins	16
2.2.3 Transcoding Publisher Toolkit	17
2.2.4 Administration	17
2.3 WTP components	17
2.3.1 Framework components	18
2.3.2 Transcoder components	19
2.4 Administrative components	20
2.5 Transcoding Publisher resources	20
2.5.1 Preference profiles	20
2.5.2 XML stylesheets	25
2.5.3 Transcoders	27
2.6 Structure of Transcoding Publisher	34
2.6.1 Source identification	36
2.6.2 Preference aggregation	37

2.6.3 How a document is transcoded	37
Chapter 3. Installation	41
3.1 Planning for Transcoding Publisher	41
3.1.1 Server configuration	41
3.1.2 Security	42
3.1.3 Hardware and software prerequisites	42
3.2 Installing Transcoding Publisher	43
3.2.1 Installation	43
3.2.2 Logging	50
Chapter 4. Configuration	51
4.1 Server configuration	51
4.1.1 Configuring Transcoding Publisher as a network proxy	53
4.2 Client configuration	57
4.3 Changing your configuration	59
4.4 Configuring Transcoding Publisher to start automatically	60
4.5 Preference profiles	60
4.6 Transcoders	61
4.7 XML stylesheet selectors	61
4.8 Administration Console	61
Chapter 5. Running as a stand-alone network proxy	63
5.1 Overview	63
5.2 Basic flow for Transcoding Publisher acting as a proxy	64
5.3 Configuration	66
5.3.1 Port mapping	66
5.3.2 Firewall configuration	67
5.4 Encryption support	68
5.5 Request Viewer	70
5.6 Troubleshooting	72
Chapter 6. Running with a caching proxy	73
6.1 Overview	73
6.2 Relationship to external cache	73
6.3 Understanding caching	74
6.4 Configuration	76
6.4.1 Considerations for configuring the external cache	79
6.4.2 When is caching beneficial?	80
6.4.3 MEG groups and how they are used	81
6.4.4 Sample configuration for Web Traffic Express	81
6.4.5 Sample configuration for WinGate	83
6.5 Tuning cache server	87
6.6 Troubleshooting	87

Chapter 7. Transcoding with WAS filters	93
7.1 Transcode at the source	93
7.1.1 Why transcode at the source	93
7.1.2 When not to use transcoding at source	94
7.1.3 Transcoding with filters	94
7.1.4 Transcoding JavaServer Pages (JSP) content	95
7.2 Configuring Transcoding Publisher as a WebSphere MIME-filter	96
7.2.1 Taking a look inside the WTP filter configuration	98
7.3 WAS servlet configuration for transcoding	98
7.3.1 XML stylesheet selection	103
7.4 Sample scenario: transcoding HTML content	107
7.4.1 Enable text/html transcoding	109
7.5 Sample Scenario: transcoding XML content	113
7.5.1 Sample TranscodingSamples	114
7.5.2 Enabling Transcoding (XML to HTML)	116
Chapter 8. Transcoding with JavaBeans	121
8.1 Overview	121
8.1.1 WTP JavaBean model	121
8.1.2 WTP-provided JavaBeans	122
8.1.3 Invoking a WTP JavaBean	127
8.2 Transform Tool sample program	127
8.3 Image transcode sample program	128
8.4 Using the WTP JavaBeans	132
8.4.1 Defining the classpath and path	132
8.4.2 Specifying input parameters	133
8.4.3 Using the JavaBeans with Linux	134
8.5 Scenario: invoking WTP JavaBeans from a servlet	135
8.6 Scenario: invoking WTP JavaBeans from JSPs	142
8.7 Scenario: invoking a WTP JavaBean to transcode XML	148
Chapter 9. Using the Toolkit	157
9.1 The Transform Tool	157
9.1.1 Starting the Transform Tool	158
9.1.2 Using the Transform Tool	158
9.1.3 Using XML stylesheets with the Transform Tool	163
9.1.4 The Transform Tool application	165
9.2 The Request Viewer	166
9.2.1 IBM WebSphere Transcoding Publisher Request Viewer	167
9.3 Creating preference profiles	174
9.3.1 Using the Create Profile wizard	174
9.3.2 Creating a network preference profile	176
9.3.3 Creating a device preference profile	184

9.4 The Snoop tool	192
9.4.1 Using the Snoop tool	193
9.4.2 Transcoding Snoop content	196
9.5 Creating new transcoders for Transcoding Publisher	197
9.5.1 Servlet resources	199
9.5.2 Adding a transcoder	199
Chapter 10. Transcoding wireless applications	207
10.1 Introduction	207
10.1.1 WAP programming model	208
10.1.2 Microbrowser	209
10.1.3 Lightweight protocol stack	209
10.1.4 Wireless telephony applications framework	209
10.2 Using Transcoding Publisher in wireless communication	210
10.3 Wireless network scenario	212
10.3.1 Components used in this scenario	213
10.3.2 Text clippers and the fragmentation transcoder	218
10.3.3 Generic HTML-to-WML transcoding	218
10.3.4 Configuration of the scenario	221
Chapter 11. Transcoding Host Publisher application content	235
11.1 Overview	235
11.2 Transcoding Host Publisher applications	237
11.3 Scenario: transcoding with WAS filters	238
11.3.1 Sample Host Publisher application	239
11.3.2 Transfer application to server	243
11.3.3 Deploy application	244
11.3.4 Defining the transcoding filter	245
11.3.5 Invoking transcoded content from a desktop browser	247
11.3.6 Transcoding HTML to WML from JavaServer Pages	249
11.4 Scenario: transcoding using WTP as a network proxy	256
11.4.1 Configuration	257
Chapter 12. Administration	261
12.1 Supported tasks	261
12.2 Using the Administration Console	261
12.2.1 The main panel	262
12.2.2 Working with resources	263
12.2.3 Working with settings	266
12.2.4 Refreshing the server	267
12.2.5 Working with logging and tracing	267
12.2.6 Starting and stopping Transcoding Publisher	268
12.3 Creating preference profiles	268
12.4 Registering preference profiles	269

12.4.1	Registering network preference profile	269
12.4.2	Registering a device preference profile.	276
12.4.3	List of preferences and their variables in profile files.	284
Chapter 13.	Problem determination	287
13.1	Typical problems	287
13.2	Tools	289
13.3	Logging and tracing	290
13.3.1	Logging	291
13.3.2	Tracing.	295
13.4	Reporting problems	299
Appendix A.	TranscodingSamples sample program	301
Appendix B.	XML stylesheet (FlightInfoForNet_IE.xsl)	305
Appendix C.	CSD Service Fix Pack - APAR IC26798.	309
Appendix D.	Special notices	311
Appendix E.	Related publications.	313
E.1	IBM Redbooks	313
E.2	IBM Redbooks collections.	313
E.3	Other resources	314
E.4	Referenced Web sites.	314
How to get IBM Redbooks	315
IBM Redbooks fax order form	316
Glossary	317
Index	321
IBM Redbooks review	323

Preface

This redbook helps you to understand the IBM WebSphere Transcoding Publisher (WTP) product. It focuses on architectures and technologies implemented in this product and helps you plan, install and configure WTP in the supported models. For example, you will find information on how to configure WTP to run as a proxy server, as a WebSphere Application Server (WAS) filter, or simply as JavaBeans within user applications.

You will also find numerous configuration scenarios showing ways to set up the IBM WebSphere Transcoding Publisher Version 1.1 to adapt and reformat your application content (HTML, XML and images) so it can be accessed from devices such as WAP phones supporting WML, Palm Pilots, Windows CE and WorkPads.

A basic knowledge of Java technologies such as servlets, JavaBeans, Java Server Pages (JSPs) and the terminology used in Web publishing is assumed.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.

Juan R. Rodriguez is a Senior Software Engineer at the IBM ITSO Center, Raleigh. He received his M.S. degree in Computer Science from Iowa State University. He writes extensively and teaches IBM classes worldwide on such topics as networking, Web technologies and data security. Before joining the IBM ITSO, he worked at the IBM laboratory in Research Triangle Park (North Carolina, USA) as a designer and developer of networking products.



Eduardo Ázara is an IBM Business Partner in Brazil. He is finishing a degree in Telecommunications Engineering at Universidade Federal Fluminense, in Rio de Janeiro, Brazil. He has six years of experience in networking pre-sales technical support with IBM. His areas of expertise include LANs, MANs, WANs, Host Publisher, Personal Communications, Communications Servers, Host On-Demand and Web protocols.



Lucia Ruiz is a System Analyst with Banco do Brasil S.A. in Brasilia (Brazil). She graduated from FATEC-SP with a degree in Computer Systems Analysis and holds a Master's degree in Object-Oriented Systems from University of Campinas (UniCamp), Campinas, Brazil. She has 10 years of experience in application development for the banking industry, including Palm Pilots, VisualAge, Java, C, C++, MQSeries, DB2, Notes, and Web technologies.



Pekka Kaukonummi is a Technical Sales Specialist for Network Computing Software and Business Communications in the Software Marketing group of IBM Finland. He started his career with IBM in 1984 as a Systems Engineer. He has held several positions as product manager, dealer representative, sales representative and IBM NCSD software representative. He graduated from the Technical Institute of Helsinki, Finland as a Telecommunications Engineer.



Thanks to the following people for their invaluable contributions to this project:

Steve Baber
IBM Research Triangle Park, North Carolina

Andrew Hately
IBM Austin, Texas

Alan Booth
IBM Research Triangle Park, North Carolina

Rick Floyd
IBM Research Triangle Park, North Carolina

Iva Anderson
IBM Research Triangle Park, North Carolina

Chris Seekamp
IBM Research Triangle Park, North Carolina

Kathryn Britton
IBM Research Triangle Park, North Carolina

Sam McHan
IBM Research Triangle Park, North Carolina

We are also very grateful to the following people that took extra time to review this redbook:

George Fridrich, Charlene Frazier, Karen Tracey
IBM Research Triangle Park, North Carolina

Richard Spinks, Jim Collins, Brian Baker
IBM Research Triangle Park, North Carolina

Shawn Walsh
IBM International Technical Support Organization, Raleigh Center

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks review” on page 323 to the fax number shown on the form.
- Use the online evaluation form found at <http://www.redbooks.ibm.com/>
- Send your comments in an Internet note to redbook@us.ibm.com

Chapter 1. Introduction

Transcoding is a new technology that gives you the ability to make Web-based information available on handheld and other new devices economically and efficiently over slow network connections such as wireless and dial up modem connections.

With transcoding, users receive information (text and images) tailored to the capabilities of the devices they are using and also tailored according to the capacity of the network being used. In this chapter we present an overview of the new IBM WebSphere Transcoding Publisher Version 1.1 product.

1.1 Overview

IBM WebSphere Transcoding Publisher makes your data work for you, when and where you need it, to help you reach and service your customers better, reduce costs, and improve productivity.

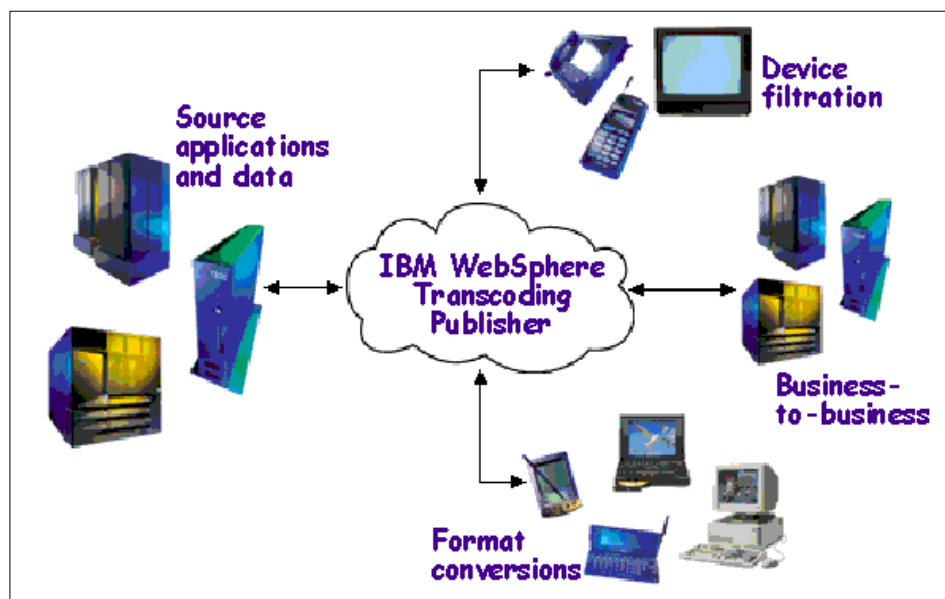


Figure 1. IBM WebSphere Transcoding Publisher enables universal access

IBM WebSphere Transcoding Publisher is server-side software that filters and reformats information that you own, or have sufficient rights to filter and reformat, from existing host and Web applications, tailoring the presentation

to be effective for different organizations, different device form factors, and different network limitations.

Transcoding Publisher provides default transformations for several leading pervasive devices. You can modify transformations based on criteria important to you or your users. For example, if your users on the road want to sacrifice aesthetics for faster download, you can tell IBM WebSphere Transcoding Publisher to reduce image file size and transform table formats to simpler list formats.

As new devices emerge in this fast-growing market, new transformations can be added easily. For specific or unique format requirements, IBM WebSphere Transcoding Publisher provides a pluggable framework for adding new device and network profiles and content type transformations.

For high level details on how IBM WebSphere Transcoding Publisher has been designed to intercept, transform requests and responses to deliver transcoded data to a specific client device, see Chapter 2, “Architecture” on page 13.

1.2 Hardware and software prerequisites

In general, IBM WebSphere Transcoding Publisher can be run in Windows NT server, Windows 2000 server, Red Hat Linux V6.1, AIX and Sun Solaris. However, you should be aware that in this release not all models and functions are supported in all platforms and you should check the product documentation for more specific information.

Hardware prerequisites are:

- Windows NT server, Windows 2000 server, and Red Hat Linux V6.1
 - Pentium 11/266 MHz or higher processor
 - 30 MB of available disk space
 - 128 MB of memory
- AIX and Sun Solaris
 - 166 MHz or higher processor
 - 30 MB of available disk space
 - 128 MB of memory

Software prerequisites are:

- Microsoft Windows NT Server (V4 with SP5) that supports JDK 1.1.8

- Microsoft Windows 2000 Server that supports JDK 1.1.8
- AIX Version 4.3.2 or higher that supports JDK 1.1.8.6
- Sun Solaris: Version 7 or higher that supports JDK 1.1.8
- Red Hat Linux V6.1 that supports JDK 1.1.8

For the Microsoft Windows NT server, AIX and Sun Solaris platforms above: If running as a servlet, WebSphere 2.03 is required. Support for WebSphere 3.x will be provided in a follow-on IBM WebSphere Transcoding Publisher service update.

National language versions of WebSphere Transcoding Publisher are supported on the Windows NT, Windows 2000, and AIX platforms.

Notice that for most platforms, the Java Development kit, JAVA Tech. Edition 1.1.8 is included with product. However, WTP does not include a JDK for the Solaris platform since it is assumed to be part of the operating system.

1.3 Preference profiles

IBM WebSphere Transcoding Publisher makes use of a standard set of device and network profiles that enable customized conversions for client devices and network services based on characteristics such as device form factor and network cost and speed. In addition, IBM WebSphere Transcoding Publisher can be extended to support your own preference profiles.

There are several profiles provided with the product and ready to use:

- Devices profiles - Windows CE devices, Palm Pilot devices, Wireless Application Protocol (WAP) phones, traditional browsers (Netscape and Microsoft Explorer) and XML-capable desktop browsers.
- Network profiles - Wireless network, dial network and network default. These profiles are only used if IBM WebSphere Transcoding Publisher is running as a Proxy.

IBM WebSphere Transcoding Publisher provides a pluggable infrastructure for sequencing transcoders, monitoring requests, resolving conversion parameters based on network and device preference profiles, and exploiting a separate Web cache. The infrastructure controls the way different transcoders work together and provides them with common services.

1.4 Transcoders

A transcoder modifies the content of a document. The transcoders provided with IBM WebSphere Transcoding Publisher include text editors and image editors.

IBM WebSphere Transcoding Publisher provides a standard set of transcoders to convert HTML documents, to reformat images, to convert XML documents to various presentation formats or other XML variants by selecting and applying stylesheets, and to convert HTML into WML.

The basic transcoders are:

1. An image transcoder that can scale, modify quality, and modify color levels in JPEG and GIF images. Additionally, the image transcoder can convert JPEGs to GIFs (for devices that do not render JPEGs) and it can also modify GIFs to JPEGs.
2. A stylesheet transcoder that selects the stylesheet and applies it to an input Extensible Markup Language (XML) document to produce a version that is appropriate for the target device.
3. A text transcoder that can modify elements of a text document based on device, network, and potentially user preference information. The primary use of this text transcoder is to modify Hypertext Markup Language (HTML) documents to remove unsupported elements, reduce space usage, replace features such as images or frames with links, and otherwise tailor documents to make them render more gracefully on constrained devices.
4. A text transcoder that can translate HTML documents to Wireless Markup Language (WML) for presentation to WAP-based smart phones.
5. A transcoder that can break up documents into fragments that are small enough to be accepted by the device. This transcoder will add links to allow the user to navigate to other fragments, as well as cache the fragments for future requests, thus managing the user's navigation around the fragmented document.

1.5 Running as a proxy

Using IBM WebSphere Transcoding Publisher as a proxy (also known as a network intermediary) is a single service that tailors content coming from many different Web servers.

The proxy intercepts HTTP requests and responses as they flow between the user and Web server.

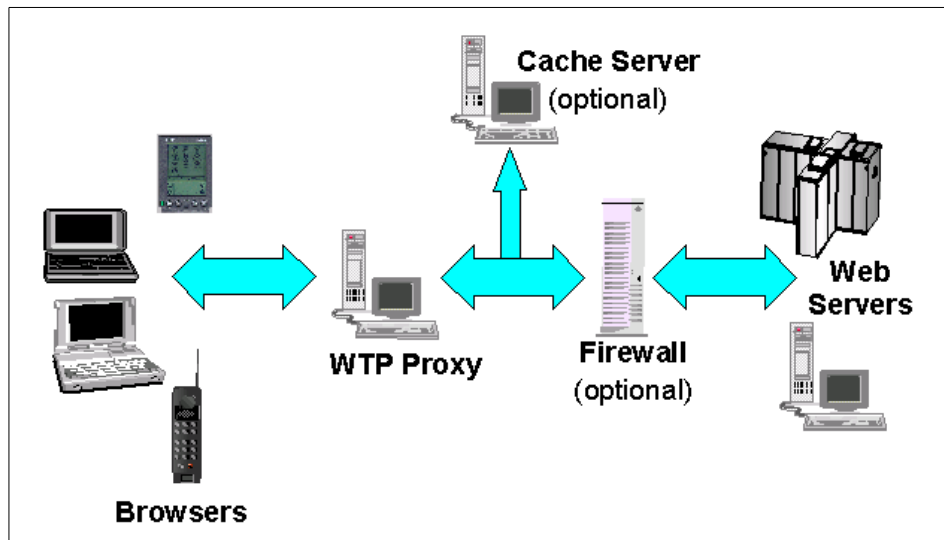


Figure 2. WebSphere Transcoding Publisher - running as a proxy

If you run IBM WebSphere Transcoding Publisher as a network proxy, and your network uses a cache server or a firewall, you will need to supply the addresses and port numbers used by these servers when you configure IBM WebSphere Transcoding Publisher.

1.5.1 Stand-alone proxy

When using the IBM WebSphere Transcoding Publisher as a normal proxy in a browser, the data that flows from the original source will be transcoded in the proxy according to the device and network profile needed. This method would be suitable for service providers and for those who already have Web pages and want to reach new users with new types of browsers. This scenario will be discussed in Chapter 5, "Running as a stand-alone network proxy" on page 63.

1.5.2 Caching proxy

If you use an external cache server in your network, IBM WebSphere Transcoding Publisher can use it to store and retrieve transcoded Web pages and intermediate results. This may enable IBM WebSphere Transcoding Publisher to avoid repeating the transcoding of frequently accessed pages, thus giving better performance.

IBM WebSphere Transcoding Publisher and the External Caching server must be within the same firewall. Expiration of cached data follows HTTP specifications. This scenario will be discussed in Chapter 6, “Running with a caching proxy” on page 73.

1.5.3 Connecting to a firewall

IBM WebSphere Transcoding Publisher (WTP) will co-operate with traditional firewalls, and when the server is installed the firewall connections will have to be configured. This configuration is introduced in 5.3.2, “Firewall configuration” on page 67.

1.6 Running as servlets (WAS filtering)

IBM WebSphere Transcoding Publisher can be configured to operate as a servlet, so that it can be administratively incorporated into the WebSphere Application Server to transcode the content produced by other servlets. The advantage of this configuration is that the transcoding servlet can operate within the security context of the Web application server so that it can transcode information that will later be encrypted before it is sent to the client.

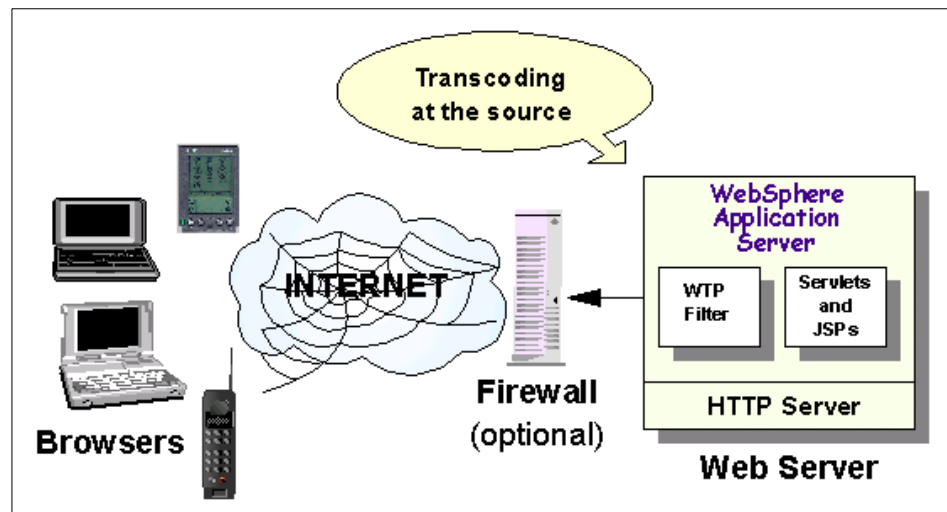


Figure 3. WebSphere Transcoding Publisher - running as a servlet (WAS filter)

Note: In this release (WTP 1.1), IBM WebSphere Application Server Version 2.03 is required to run IBM WebSphere Transcoding Publisher as a servlet filter. In addition, when you run Transcoding Publisher as a filter, you cannot

use network preference profiles or WML deck fragmentation. It is expected that WTP will be enhanced in future releases to provide extended support.

This configuration is not supported on Linux. See 1.2, “Hardware and software prerequisites” on page 2 for supported platforms.

For more details about how IBM WebSphere Transcoding Publisher is configured to run as a WAS filter, see Chapter 7, “Transcoding with WAS filters” on page 93.

1.7 Running as JavaBeans

It is also possible to separate the transcoders from the framework and run them independently as JavaBeans. This provides a means for other server programs, such as servlets, independent content-providing programs, or JavaServer Pages (JSP), to invoke single transcoders directly. The transcoders provided as JavaBeans are listed in Table 1.

Table 1. WTP provided transcoders as JavaBeans

Transcoder	Function
ImageTranscoderBean	Converts images
HtmlReducerBean	Simplifies HTML documents
HtmlHandlerBean	Converts HTML to WML
XmlHandlerBean	Converts HTML to XML

For details about how IBM WebSphere Transcoding Publisher can be invoked as JavaBeans see Chapter 8, “Transcoding with JavaBeans” on page 121.

1.8 Administration Console

The Administration Console (graphical interface) is a Java application that is used to perform configuration, administration and tracing tasks, as well as integrating and configuring new profiles, stylesheet selectors, and transcoders.

The IBM WebSphere Transcoding Publisher Administration Console provides (see Figure 4):

- A menu structure to provide access to other tasks
- A tree view of your resources/objects (preference profiles, transcoders, and stylesheet selectors)

- The object view, with the information related to the tree node selected
- The ability to edit individual resources
- Help for each panel, as well as a list of How do I ... topics

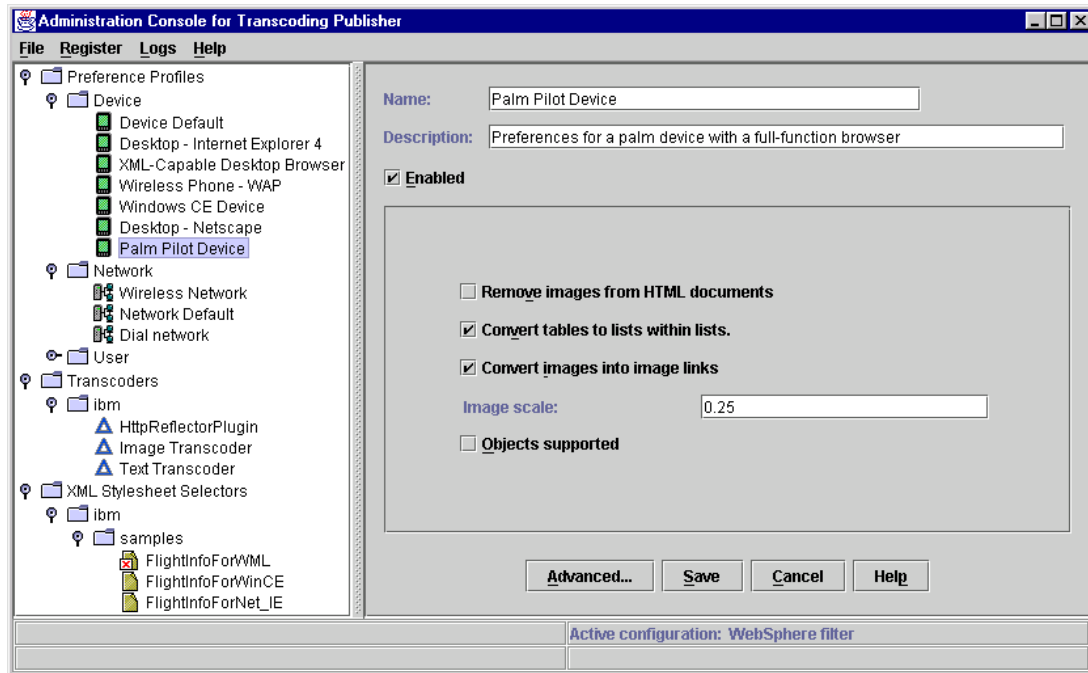


Figure 4. WTP Administration Console

For details about IBM WebSphere Transcoding Publisher Administration Console see Chapter 12, “Administration” on page 261.

1.9 Toolkit

The developer Toolkit is intended for device manufacturers, ISVs, and service people who want to extend the existing behavior of IBM WebSphere Transcoding Publisher.

The IBM WebSphere Transcoding Publisher Toolkit consists of the following:

- Information (documentation and samples) that helps developers understand WTP capabilities and extend IBM WebSphere Transcoding Publisher to support new uses and data models
- Tools that aid deploying framework extensions for example:

- a. The Create Profile option to create new preference profiles
- b. A procedure to create new transcoders
- c. The Transform Tool for testing transformations
- d. The Request Viewer to monitor the dataflow through IBM WebSphere Transcoding Publisher configured as a proxy.

For details about IBM WebSphere Transcoding Publisher Toolkit see Chapter 9, “Using the Toolkit” on page 157.

1.10 Integration with Host Publisher

Transcoding can be used together with Host Publisher applications to extend the use of legacy information to pervasive devices. If you want to transcode a Host Publisher application, you can use any of the scenarios described in this book.

To use with the proxy configuration, you do not need to change the application or the Host Publisher Server configuration. See 11.4, “Scenario: transcoding using WTP as a network proxy” on page 256 for details.

If you want to use filters, you have to configure WebSphere Application Server to use the WebSphere Transcoding Publisher filters (this is described in more detail in Chapter 11, “Transcoding Host Publisher application content” on page 235).

Or you can change the application in order to call the transcoders (JavaBeans) provided with WebSphere Transcoding Publisher to transcode that application only. This configuration is not described in this book but you can read more about WTP JavaBeans in Chapter 8, “Transcoding with JavaBeans” on page 121.

1.11 Tracing and logging

WebSphere Transcoding Publisher (WTP) provides two tools for diagnosing problems: the tracing and logging facilities in the WTP Administration Console as illustrated in Figure 5.

The trace facility allows you to see three message levels:

- Low level includes errors and exceptions only.
- Medium level includes Low plus considerable information about individual requests.

- High level includes Low, Medium and exhaustive information about requests.

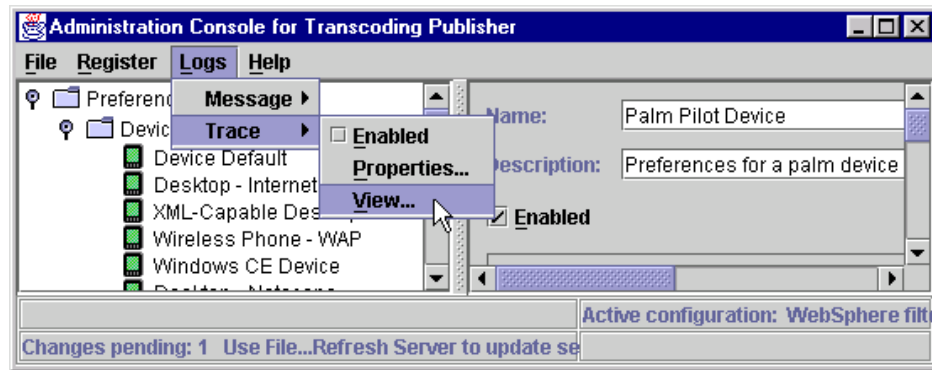


Figure 5. WTP Administration Console - message logging and trace capabilities

The logging facility allows you to see information, warning and error messages. Information messages record normal events. Warning messages indicate a possible problem. Error messages indicates a definite problem requiring administrator attention.

For more details about the WTP message and trace facility see Chapter 13, “Problem determination” on page 287.

1.12 National language support

IBM WebSphere Transcoding Publisher Version 1.1 is available for the following languages:

- English
- French
- German
- Italian
- Spanish
- Brazilian Portuguese
- Canadian French
- Simplified Chinese
- Traditional Chinese
- Japanese

- Korean

Note: The online documentation files (help, Administrator's Guide, Developer's Guide) for Transcoding Publisher are stored in ten language directories under doc/source. Each language directory takes up about 800KB of disk space. If you are concerned about disk space, you can delete the directories for the languages you do not use. Deleting all but one will save about 7MB of disk space.

1.13 WTP 1.1.1 CSD

IBM WebSphere Transcoding Publisher Version 1.1.1 is now available for download from the WTP service site:

<http://www6.software.ibm.com/enetwork/isd/home.html>

The sample scenarios described in this redbook have been implemented using this service fix pack. For information on what problems have been fixed in this CSD via APAR IC26798 see Appendix C, "CSD Service Fix Pack - APAR IC26798" on page 309.

1.14 Resources used in this redbook

The following resources have been used during the development of this redbook:

- Online product documentation:
 - a. Administrator's Guide
 - b. Developer's Guide
 - c. Readme File
 - d. JDK 1.1.8
 - e. Toolkit Readme file
- IBM WebSphere Transcoding Publisher on the Internet:
<http://www.ibm.com/software/websphere/transcoding>
- IBM Host Publisher on the Internet:
<http://www.ibm.com/software/network/hostpublisher>
- Host Publisher User's Guide (product online documentation)
- *IBM WebSphere Performance Pack: Caching and Filtering with IBM Web Traffic Express*, (SG24-5859)
- WapForum on the internet: <http://www.wapforum.org>

- Nokia WAP Phone Emulator: <http://www.nokia.com/wap/index.html>
- WinGate on the Internet: <http://www.wingate.com>

Chapter 2. Architecture

IBM WebSphere Transcoding Publisher is network software that modifies content presented to users based on the information associated with the request, such as device constraints, network constraints, user preferences, and organizational policies. In this chapter we present a high level overview of how IBM WebSphere Transcoding Publisher has been designed to accomplish this.

2.1 Overview

Transforming content can reduce or eliminate the need to maintain multiple versions of data or applications for the following categories:

- Device types such as desktop browsers, WAP phones, WML devices, palm devices and so on.
- Network service levels such as dial and wireless networks.
- User preference. However, in this release, IBM WebSphere Transcoding Publisher Version 1.1 only provides a default user category, which will be extended in a future release.

IBM WebSphere Transcoding Publisher transforms the data by intercepting and transforming requests and responses to deliver a targeted response to a user. IBM WebSphere Transcoding Publisher tailors HTTP requests, generates, transforms and filters the responses.

IBM WebSphere Transcoding Publisher helps you ensure that Web data delivered to a device is appropriate for the device, the network, and your administrative policy in terms of:

- Size - scaled presentation of images and compression of text, which could, for example, reduce the size of files that are downloaded
- Detail - specifically tailored to the amount and kind of information your users need
- Format - data presented in the format supported by your users' device types

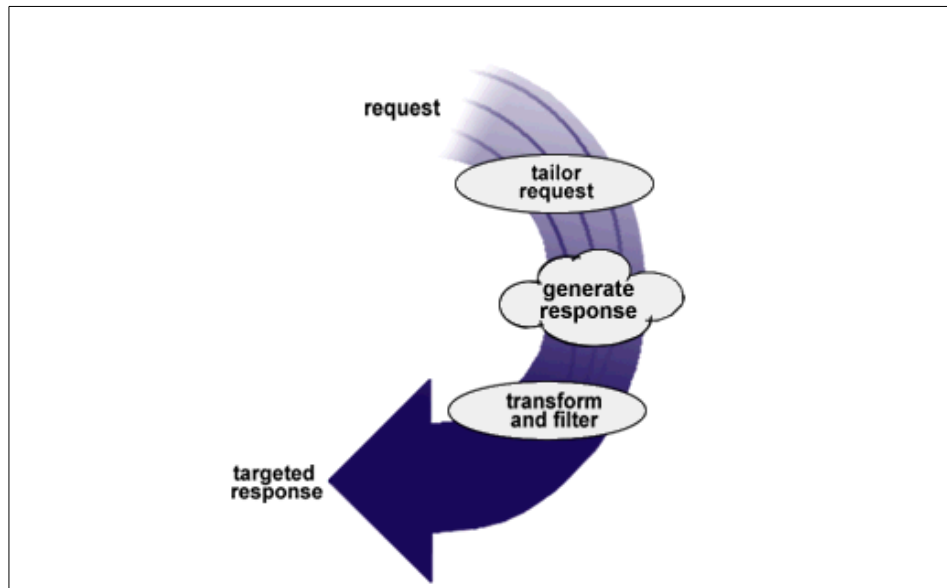


Figure 6. IBM WebSphere Transcoding Publisher - content transformation

2.2 Transcoding structure

As shown in Figure 7 on page 15, IBM WebSphere Transcoding Publisher contains:

- A pluggable framework that hosts custom and IBM-provided transformation plug-ins, or transcoders. New transcoders can be added and can interact with existing transcoders. All plug-ins can leverage a set of core services, such as the ability to acquire preference information in order to respond differently to requests for different networks or different devices.
- A base set of transcoder plug-ins that transform Web content. For example, one of the transcoders can select and apply the appropriate Extensible Stylesheet Language (XSL) stylesheet to transcode an Extensible Markup Language (XML) document for rendering on a particular device. The framework can also host transcoders for other purposes, such as personalizing Web pages, transcoding printable documents for Web viewing, and converting from legacy formats such as AFP (Advanced Function Presentation) to Internet formats such as the World-Wide Web Consortium (W3C) Scalable Vector Graphics (SVG).

- A developer's toolkit to support building custom transcoder plug-ins. Custom transcoders can be used to process additional data formats, to extract the most important elements of a full screen application for display on a very small screen, or to improve the transcoding associated with specific Web applications.
- Administration services to give administrators control over configuration information and preference profiles. Administrators can also view and control message and trace logging.

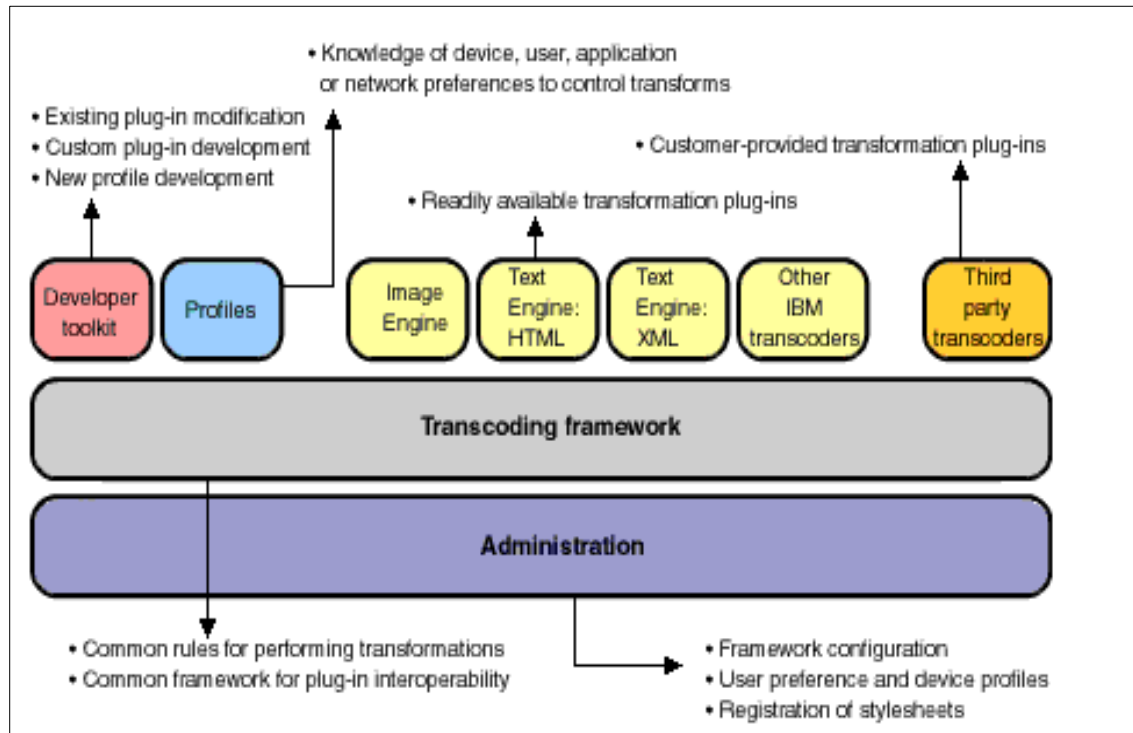


Figure 7. IBM WebSphere Transcoding Publisher - architecture overview

For more information visit the WBI SDK Web site:

<http://www.almaden.ibm.com/cs/wbi/doc/index.html>

2.2.1 Pluggable framework

In a typical Web environment, information is simply sent from a server to a browser for display and interaction. However, there are many ways that adding an intermediary between the browser and the server can improve the system. For example, an intermediary can keep track of the information the

user has viewed in order to make it easier to find information again. Or, an intermediary may enhance the information the user sees by adding annotations and personalization beyond what the server was designed to do. Intermediaries turn the network into a "smart pipe" with applications that can enhance the information on the Web.

The pluggable framework incorporates the Web Intermediaries (WBI) Development Kit.

The extensible WBI intermediary backbone is combined with transcoding applications to produce a Web proxy that can be used to transform images and text according to user, network, and device preferences. The WBI technology can host a number of different transcoding applications that can modify requests on the way to the Web server or responses on the way back to the client. For each transaction, WBI selects and sequences the correct set of transcoders to produce the desired result for the specific request.

2.2.2 IBM Transcoder plug-ins

The basic Web-oriented transcoder plug-ins that come with IBM WebSphere Transcoding Publisher include:

- The image engine, which transcodes GIF and JPEG images by converting between formats and modifying scale and color level. The image engine will be extended to handle other image formats in future releases. This transcoder uses the Jimage library developed by IBM Research.
- The text engine, which does various types of text transcoding:
 - Simplification of HTML documents. For example, the text engine can replace images embedded in pages with links to images and remove objects or features, such as animated images or JavaScript, that are not supported by the target device.
 - Stylesheet application to XML documents. This transcoder uses the open source Xalan package available from Apache, with enhancements for selecting an XSL stylesheet based on the application, user, device, or network associated with the specific request. For details see <http://xml.apache.org/xalan/overview.html>.
 - Format translation from HTML to Wireless Markup Language (WML). This can be the first step in modifying a browser-based application for display on a client device enabled for the Wireless Application Protocol (WAP). A custom transcoder to select the right subset of the application content is also needed for effective HTML-to-WML transcoding. Customers can use the transcoding developer's toolkit to produce the

custom transcoders. For more information see 9.5, “Creating new transcoders for Transcoding Publisher” on page 197.

The text engine incorporates XML, stylesheet, and text technologies.

- IBM XML Fragmentation transcoder fragments documents into pieces small enough to be managed by the receiving device.
- Resource repository stores multiple versions of transcoded resources. In this release, the fragmentor is WML-specific. The resource repository is used by the fragmentor to store the generated fragments.

2.2.3 Transcoding Publisher Toolkit

IBM WebSphere Transcoding Publisher Toolkit contains a set of samples, instructions, documentation, and procedures to allow the development of customized transformations and to enhance the profiles used to control transformations. The toolkit also includes the Transform Tool which can be used to test the result after transcoding in source level or when used with image files. It has also the Request Viewer which can be used to monitor the operation of transcoders installed in IBM WebSphere Transcoding Publisher Server, running as a proxy.

2.2.4 Administration

The Administration Console in IBM WebSphere Transcoding Publisher allows you to do the following administrative services:

- Register new preference profiles, transcoders and XML stylesheets.
- Modify values in preference profiles and stylesheet selectors.
- Enable or disable transcoders, profiles and stylesheet selectors.
- View messages such as log entries as explained in Chapter 13, “Problem determination” on page 287.
- Configure and view trace information as explained in Chapter 13, “Problem determination” on page 287.

For more information about how you can use these services see Chapter 12, “Administration” on page 261.

2.3 WTP components

In this section we briefly explain the different components in the IBM WebSphere Transcoding Publisher product:

- Framework components

- Transcoder components
- Administrative components

2.3.1 Framework components

Pluggable APIs allow new transcoding plug-ins to be added. This allows the transcoding proxy to be extended to support new data formats or to meet special transcoding needs.

The WBI API is documented in the WBI Development Kit. Plug-in components written to this API are generally called MEGs, for "Monitor", "Editor", or "Generator" (for details see <http://www.almaden.ibm.com/cs/wbi>).

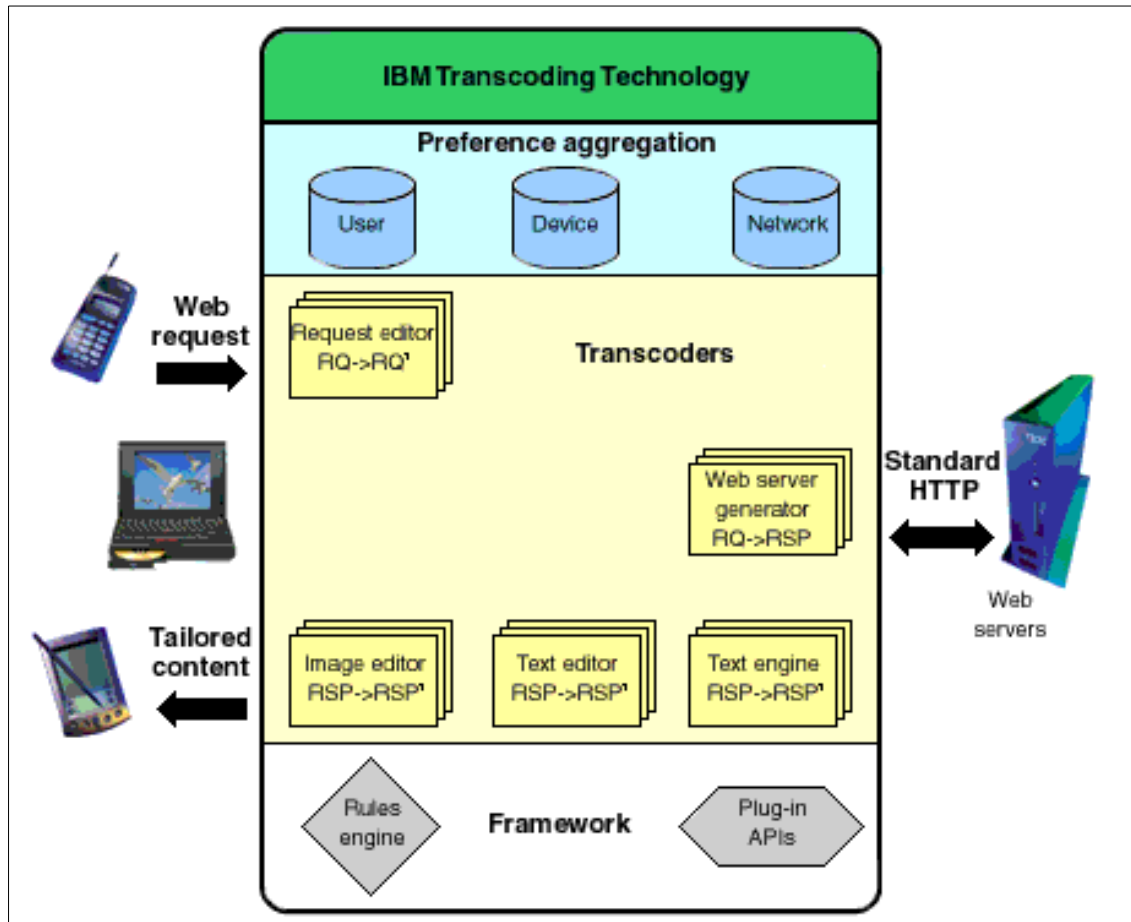


Figure 8. IBM WebSphere Transcoding Publisher components

A transcoder can include one or more of the following types of MEGs:

- Request editors that modify information in the request itself, such as fields in the HTTP header. A request editor is shown in Figure 8 on page 18 as a request transform RQ->RQ'.
- Generators that convert a request into a response. A Web server generator is illustrated in Figure 8 on page 18 as RQ->RSP. The most common example is a generator that gets an object from a Web server.
- Editors that modify the response. A response editor is shown in Figure 8 on page 18 as RSP->RSP'. Editors can customize the response on its way back to the browser.
- Monitors observe a response without changing it (RSP->RSP). They can do things like count accesses to particular pages.

The transcoding framework also allows industry-standard servlets to be incorporated as transcoders. Servlets written for other environments can be easily ported to run in the proxy, and new plug-ins written to the Java Servlet API will be usable in the many Web servers that support servlets.

The transcoding proxy supports Version 2.1 of the Sun Java Servlet API. Servlets written to this API should run with few or no changes in the transcoding framework. Plug-in components written to this API are generally called MEGlets, because they play the same roles as MEGs, but are written to the servlet API. The rules engine (see Figure 8 on page 18) is the core of the WBI run-time services. This component uses rules based on information from the request and response such as the request URL and the response MIME type for selecting the transcoders to be run and arranging them in a specific sequence.

The preference aggregator as illustrated in Figure 8 on page 18 resolves potentially conflicting values of the same preference from different sources, giving the system the appearance of a single preference value. For example, the device profile can state that color images are supported, while the network profile can specify that color should be suppressed to save network bandwidth. The preference aggregator resolves the conflict, giving the transcoders a single answer for this preference.

2.3.2 Transcoder components

The image engine includes several MEGs as follows:

- A request editor to update the Accept field in the HTTP request. For example, assume that the browser on a particular device can only show GIF images, but it does not properly set the HTTP header fields to indicate

that it can display GIF images. Some Web servers will, therefore, not send GIFs.

A request editor in the image engine can fix the HTTP header to show that the device supports GIFs. It can also modify the HTTP headers to show the device supports JPEG images, because an editor in the image engine can convert JPEGs to GIFs as the response comes back.

- An editor that modifies the scale and quality of GIF or JPEG images returned by the Web server. This editor can also convert GIFs to JPEGs or JPEGs to GIFs, as needed by the specific device preference profile.

The text engine includes several MEGs as follows:

- A generator that recognizes rewritten URLs from previous requests and processes them appropriately. For example, this generator responds to "convert images to links" Web addresses (URLs) sent down in earlier responses. This generator decodes the URL, saves information about scale or placement, and gets the image from the Web server.

The information saved from the URL is then available to other transcoders that later work on the response.

- An editor that processes HTML pages to modify the elements in the page according to resolved user, network, and device preferences.
- An editor that selects the stylesheet for the particular response and invokes the Lotus XSL tool to apply the stylesheet.
- An editor that translates common HTML elements into WML.

2.4 Administrative components

They support the mechanism used by the preference aggregator to acquire preferences. Preference profiles can be associated with device types and network access types. In a future release they will also support users or user defaults and application Web addresses.

2.5 Transcoding Publisher resources

IBM WebSphere Transcoding Publisher uses three types of resources to process documents. This section describes each resource and how it is used.

2.5.1 Preference profiles

IBM WebSphere Transcoding Publisher uses preference profiles to represent the characteristics of devices and networks, and a default user profile to

represent organizational policies. Each profile defines how IBM WebSphere Transcoding Publisher should treat documents delivered to that device or over that network.

A preference profile can represent a particular type of device, such as a WorkPad, or a particular network type, such as a wireless network. For example, on a wireless network, you might want to specify that images should be converted to text links, allowing the page to load faster. On devices with limited capabilities, you might want to convert tables to lists to reduce the horizontal spread of the display.

When IBM WebSphere Transcoding Publisher processes a document, it selects a network profile and a device profile to apply to the document. The network profile is determined by the port number on which IBM WebSphere Transcoding Publisher received the request. The device profile is determined by matching the value of the User-Agent keyword in the HTTP header of the document to one of the user-agent values defined to WTP by registering a device profile. If the two preference profiles contain preferences that conflict, for example, if they specify different image scale factors, the default preference order is:

1. User
2. Specific network
3. Specific device
4. Default user
5. Default network
6. Default device

The above search order is used unless there is a specific ordering instance. Currently there is one for WML devices and for the value `disposelImages`:

1. Device
2. User
3. Network
4. Default device
5. Default user
6. Default network

If a value is not specified for a preference in one profile, IBM WebSphere Transcoding Publisher will work down the list till it finds a value. The transcoders that will be applied to the document are selected based on the combined profiles.

IBM WebSphere Transcoding Publisher provides preference profiles for several common pervasive devices and for three network types. There are

default preference profiles to be used if none of the existing preference profiles matches the device or network being used. These preference profiles are listed in the tree area of the Administration Console. You can view the details of a preference profile by selecting it from the tree. The details will be displayed in the panel to the right of the tree view. You can modify the profile if you wish.

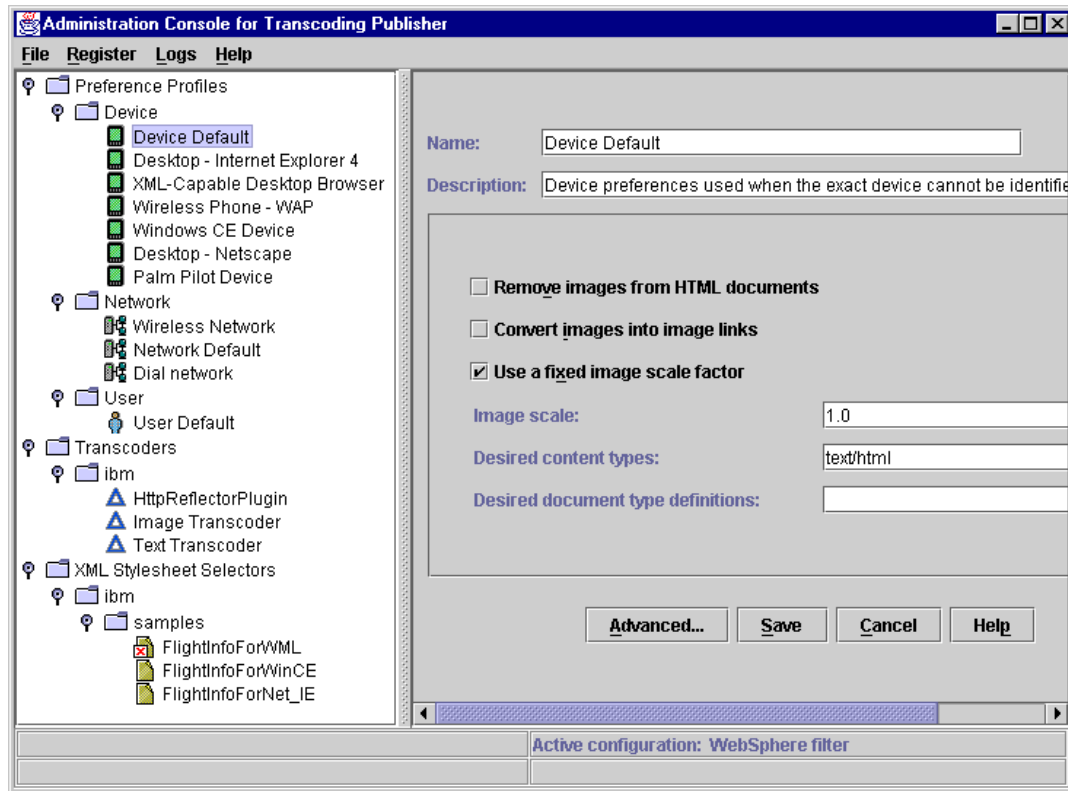


Figure 9. Preference Profiles in the Administration Console

In addition to determining which transcoders will be used, a profile can be configured to select a stylesheet to be used with XML documents.

It is also possible to modify an existing preference profile to change how IBM WebSphere Transcoding Publisher handles documents requested by a particular device or network. For example, you might decide that you want to change image processing to favor download speed over image quality more than the profile currently specifies.

Since IBM WebSphere Transcoding Publisher provides a framework, this allows you to add more capabilities to meet your needs for transcoding Web content. For more information see Chapter 9, “Using the Toolkit” on page 157. In addition, the Developer's Guide describes information on how to create new preference profiles, XML stylesheets, and transcoders. You may also acquire new resources from IBM, a device manufacturer, or another source.

IBM Transcoding Publisher will recognize devices based on the value in the User_Agent field. The backbone uses the *UserAgentToDeviceMap.prop* file located under *etc/config*. One device can have multiple string values for the User_Agent field like *WAP* or *Nokia* etc.



Figure 10. Creating a new device preference profile - user agent field definition

Network preference profiles are recognized by the port value. Network preference profiles are usable only in the proxy model or caching proxy model, because the port mapping information is not available for the transcoding servlet or JavaBeans.

Register a Preference Profile

The Preference Profile for the network must be associated with a port through which the network traffic will be transcoded by Transcoding Publisher.

These ports are already in use by Transcoding Publisher:

8089 8088 8090

What port number would you like to use?

any port number that is not in use

Description
Transcoding Publisher uses the port number to identify traffic from this network type. Ensure that the value is different from the port numbers associated with other network profiles and that it does not conflict with ports already in use by other programs on your server.

< Back Next > Finish Cancel

Figure 11. Setting up the port number when registering a network profile

Note

In this release of Transcoding Publisher it is only possible to create new profiles. That is, you cannot modify any of the existing profiles.

When you create a new profile, we suggest the following procedure:

1. Select all options for *include* and *configurable* and edit the Default values. See Figure 12.
2. Use the Administration Console to register and test the profile to work the way you want. This allows you to change or update the way it transcodes.
3. When done, create another profile with the options you have selected on the first one. Then you can select what options you will set as *configurable*.

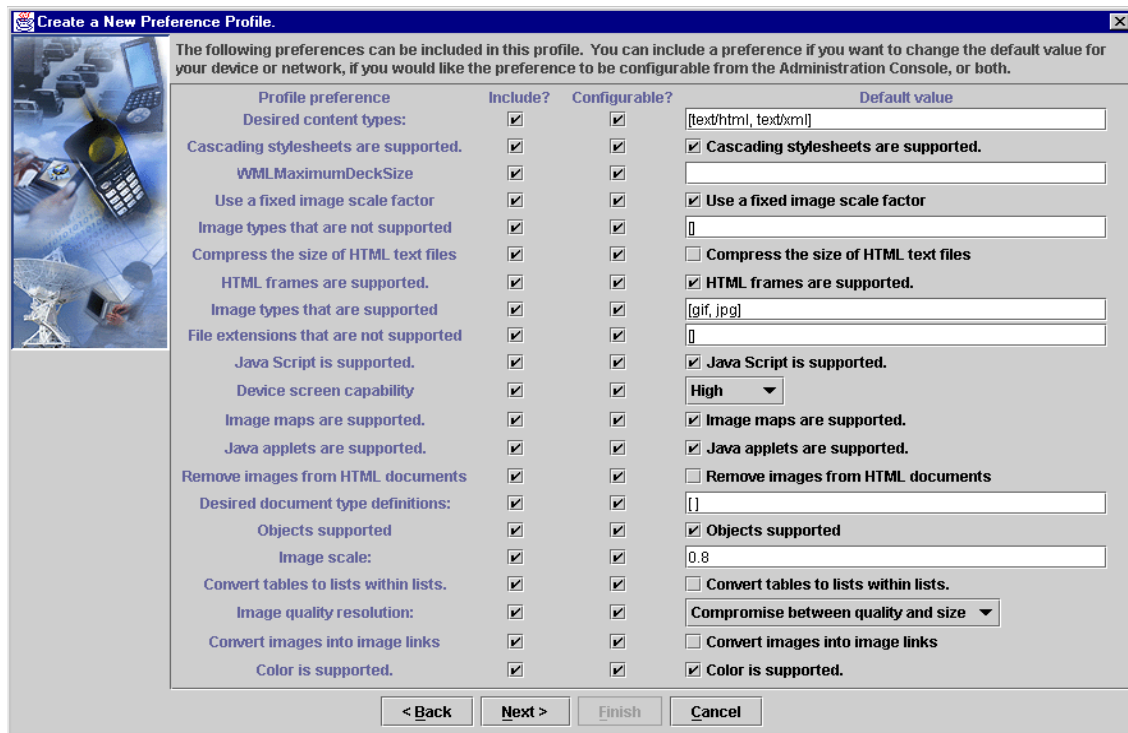


Figure 12. Creating profiles with all options selected

For more information about creating and registering preference profiles see 9.3, “Creating preference profiles” on page 174 and 12.4, “Registering preference profiles” on page 269 respectively.

2.5.2 XML stylesheets

When Transcoding Publisher processes documents composed in Extensible Markup Language (XML), it uses XSL stylesheets to convert these documents. XML documents can be converted to any markup language, including:

- A different dialect of XML, as defined by an output DTD (document type definition)
- HTML, for display on a Web browser
- WML (wireless Markup Language), for display on devices using WAP (Wireless Application Protocol).
- VoiceXML to be sent to a speech generation engine (supported with stylesheet methodology).

- HDML to be sent to North American smart phones (supported with stylesheet methodology).

Each stylesheet is represented by a stylesheet selector, which contains conditions for the selection of the stylesheet for use by IBM WebSphere Transcoding Publisher. Stylesheet selectors are listed in the tree view of the Administration Console. You can organize them into folders for easy access.

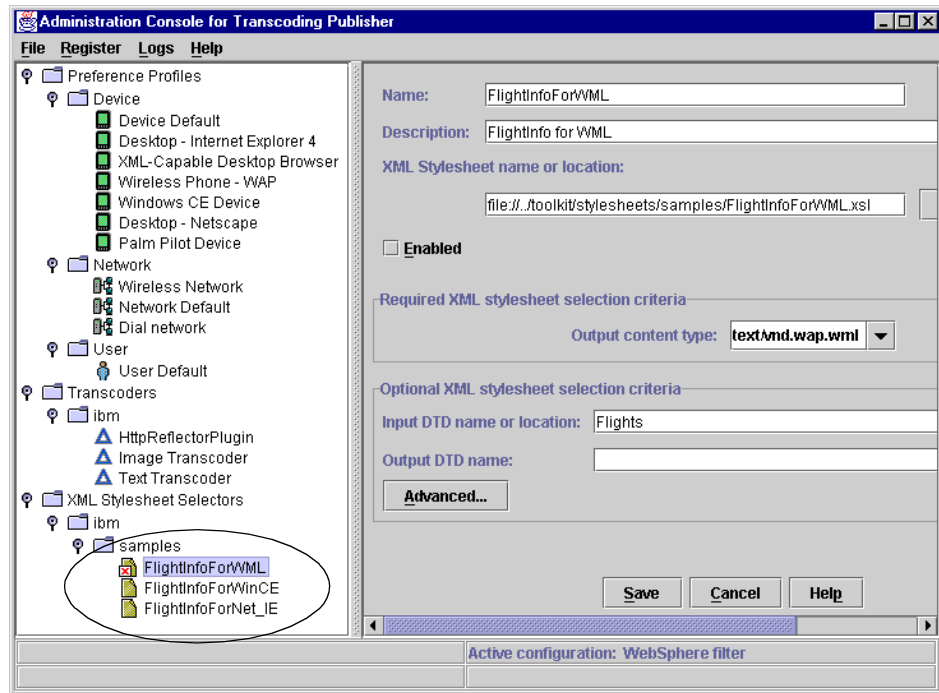


Figure 13. XML stylesheets

These stylesheet selectors can also be modified. The selection criteria defined for this stylesheet will determine the XML documents for which this stylesheet will be used. Each stylesheet must have a different set of selection criteria.

A stylesheet will be selected only if all of the selection criteria defined for it are true. If there are two or more stylesheets whose document-related criteria are all satisfied by a particular XML document, a stylesheet may be chosen by its association with preference profiles in effect when the document is processed. Each stylesheet selector must include at least the one required selection criterion, which is the output content type.

For a sample scenario of transcoding XML documents see 7.5, “Sample Scenario: transcoding XML content” on page 113.

2.5.3 Transcoders

A transcoder is a program that modifies the content of a document. The transcoders included with IBM WebSphere Transcoding Publisher can be divided into two types:

- Text editors, which translate text from one tagging language to another or modify the tagging within a language. Text editors make changes such as changing tables to lists and changing paragraph tags to line breaks.
- Image editors, which change images from one type to another (JPEG to GIF) or change the screen size, file size, or other characteristics of images contained within the document being processed. The image engine transcodes common image formats JPEG and GIF to perform:
 - Format conversion
 - Scaling
 - Conversion to grayscale
 - Quality reduction (JPEG) or color depth reduction (GIF)

The image engine determines the following information:

- Output formats supported
- Screen pixels
- Whether the device is monochrome or color
- Color depth

One special text editor is called the XMLHandler. This transcoder is responsible for selecting and applying stylesheets when XML documents are processed.

Several transcoders are provided with Transcoding Publisher, and you can obtain or develop others as well.

Table 2. Transcoders and function they provide

Transcoder	Function
Image transcoder	Modifies size and quality of GIF and JPEG images
Text transcoder	Modifies HTML or XML documents

Transcoder	Function
IBM XML Fragmentation Transcoder	Fragments WML documents into pieces that can be managed by the receiving device
Resource repository	Stores multiple versions of transcoded documents
Servlet API	Hosts plug-ins that are built to the servlet API

Some of the transcoders provided with Transcoding Publisher are also provided as JavaBeans, which can be run independently of the Transcoding Publisher transcoding server. This provides a means for other server programs, such as servlets, independent content-providing programs, or JavaServer Pages (JSP), to invoke single transcoders directly. IBM WebSphere Transcoding Publisher JavaBeans are explained in Chapter 8, “Transcoding with JavaBeans” on page 121.

The JavaBean wrapper provides the transcoder with the same information about the system and request that it receives inside the transcoding server, so that it operates the same way in both contexts.

2.5.3.1 HTML reduction

HTML reduction is used to modify HTML to something more suitable for the requesting device or based on the network preferences. So the requests from different device types can be transcoded differently from the same source. Also if the throughput of the network is low the images may not be valid to send etc.

For the device:

- Display limitations
 - Smaller screens
 - No color support
 - Limited image rendering support
- Devices with limited-function browsers
 - No scripts
 - No frames
 - No tables

For the network the throughput might be limited and then:

- Don't send images

- Generate links to images
- Compress the source

This function can reduce the amount of data being sent to the requesting device by removing images, attributes on tags, tags, scripts etc.

This function simplifies the HTML constructs used like replacing tables with lists, removing frames etc. It cannot add ability to a limited device to execute applets or fix malformed HTML.

2.5.3.2 Convert HTML in Wireless Markup Language (WML)

The most modern way to access Web content is from a WAP (Wireless Application Protocol) device. These are used in consumer markets and also in business to enable more users access to the source of information like travel information, weather information, stock information etc. Today's Web information is built for standard desktop Web browsers with the capability to show images, color, frames etc. This is not applicable for smaller devices like WAP phones. To make those Web pages usable in WAP devices we need a conversion to WML.

WML is much more simple than original HTML and also the content of HTML is often too much for a WAP device with limited memory capability. The conversion will use a parser that converts the HTML to WML. The amount of data might still be too much for the device and a text clipper can be used to remove all but the most relevant information. This is done on a page-by-page basis.

Text clipping is possible by working with a text stream or with the Document Object Model (DOM). HTML is parsed first into DOM. In the toolkit there are two sample text clippers. One is for the text stream model and one for the DOM. Text clippers are normally used for WML but are not limited to WML. After clipping the output still might need some fragmentation to fit into the WAP device properly and this is done using the fragmentation engine.

For more information about text clipping see 10.3.1.2, "Text clippers" on page 214.

2.5.3.3 Apply stylesheets to XML documents

XML (Extensible Markup Language) represents the data only and not the presentation. It is a common data language for multiple types of output and used primarily for business-to-business communication. Stylesheets are an easy way to render different presentations of XML. The text engine supports

the use of XSLT stylesheets to transform XML. Lotus XSL is used to apply stylesheets.

The text engine determines which stylesheet to apply using:

- Original XML document
- Preferences
- Configuration information about the stylesheets

The reason to use IBM WebSphere Transcoding Publisher (WTP) to apply stylesheets include the following:

- WTP helps in organizing stylesheets.
- WTP gives flexibility in configuring stylesheets
 - Associate with device type
 - Associate with URLs of documents they process
 - Associate with arbitrary key/value pairs
- WTP selects the stylesheet to apply.

There are some things that cannot be done with this function:

- Apply multiple stylesheets to one document.
- Choose the best stylesheet if multiple stylesheets are configured identically.
- Apply stylesheets to something other than XML.
- Generate XML from another markup language.

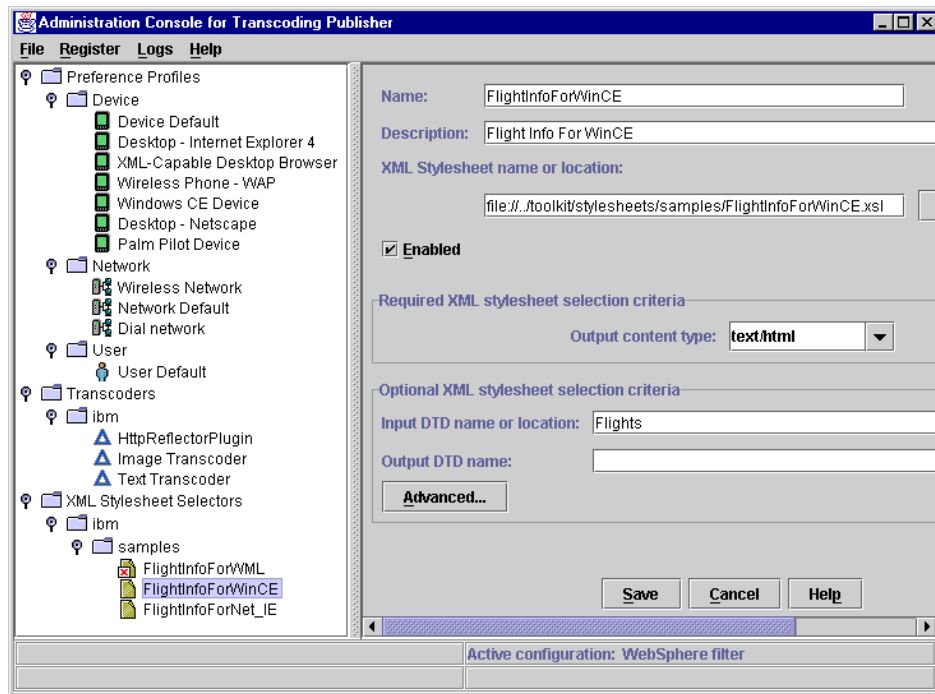


Figure 14. Sample stylesheet selector for Windows CE

2.5.3.4 Image engine

Images that appear well on desktop machines may take too long to download and they might be too large for PDA-sized device screens or when slow networks are used. The image engine helps to address all these by performing image scaling, format conversion and quality reduction.

As with other transform engines, the device profile selected for the particular request contains parameters that affect image transcoding. The device preference profile contains information on device capabilities, such as supported image formats and color or monochrome display. In addition, the device preference profile contains administrator-settable preferences such as the desired image scaling factor.

The network profile selected for the request contains parameters that affect image transcoding. The only image engine preference in a network preference profile concerns the tradeoff between image size (in bytes) and quality. It can be set to favor small size, favor high quality, or compromise between the two. For low bandwidth networks this would probably be set to favor small size or compromise.

Some browsers/devices send other HTTP headers that may be used to determine the transform. Specifically, the Windows CE browser sends UA-color and UA-pixels, which describe the screen capabilities and size. If present these headers are used in determining the transform.

This is the device preference profile for a Palm Pilot. The only configurable image engine preference is the image scaling factor, which is shipped with a default setting of 0.25. In addition the profile contains the following non-configurable device characteristics:

- supportedImages: gif
- fixedImageScale: true
- colorSupported: false
- screenCapability: low

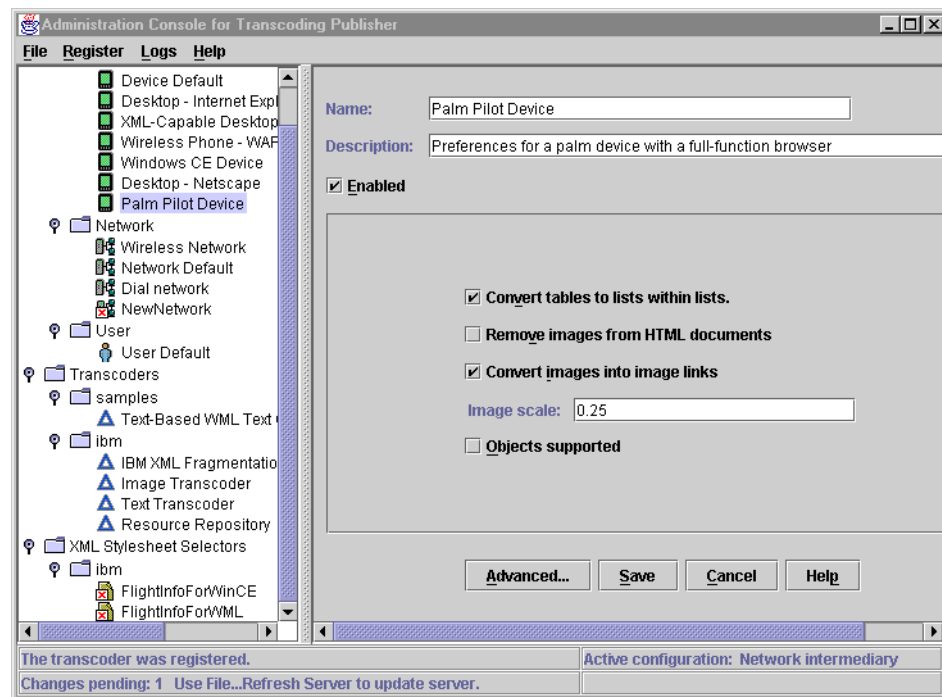
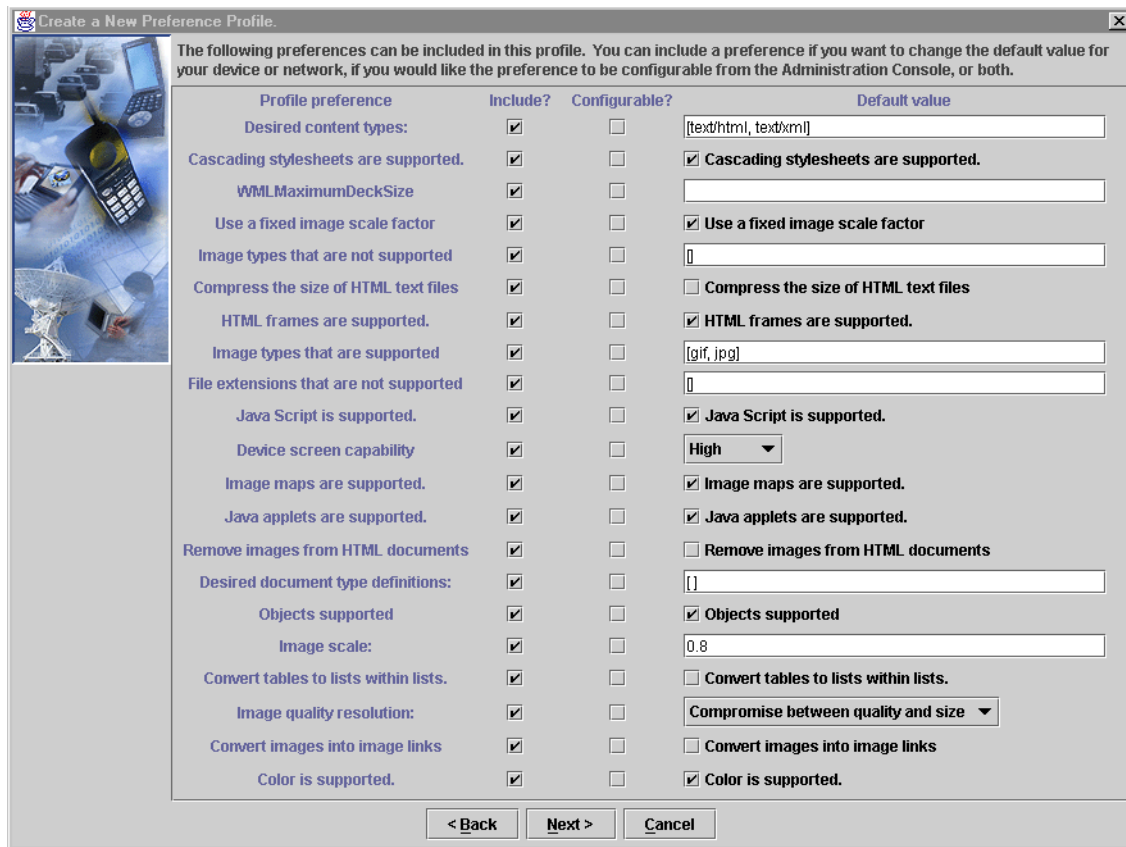


Figure 15. Sample Palm Pilot device profile - graphics settings

There are various parameters that are used for image transcoding. Some of these parameters are modifiable by the administrator, and therefore show up in the administration console. Other parameters are based on device limitations and therefore cannot be modified by the administrator. When a

new device or network preference profile is defined, the basic rules for the image engine will be selected.



Create a New Preference Profile.

The following preferences can be included in this profile. You can include a preference if you want to change the default value for your device or network, if you would like the preference to be configurable from the Administration Console, or both.

Profile preference	Include?	Configurable?	Default value
Desired content types:	<input checked="" type="checkbox"/>	<input type="checkbox"/>	[text/html, text/xml]
Cascading stylesheets are supported.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> Cascading stylesheets are supported.
WMLMaximumDeckSize	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Use a fixed image scale factor	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> Use a fixed image scale factor
Image types that are not supported	<input checked="" type="checkbox"/>	<input type="checkbox"/>	[]
Compress the size of HTML text files	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> Compress the size of HTML text files
HTML frames are supported.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> HTML frames are supported.
Image types that are supported	<input checked="" type="checkbox"/>	<input type="checkbox"/>	[gif, jpg]
File extensions that are not supported	<input checked="" type="checkbox"/>	<input type="checkbox"/>	[]
Java Script is supported.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> Java Script is supported.
Device screen capability	<input checked="" type="checkbox"/>	<input type="checkbox"/>	High
Image maps are supported.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> Image maps are supported.
Java applets are supported.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> Java applets are supported.
Remove images from HTML documents	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> Remove images from HTML documents
Desired document type definitions:	<input checked="" type="checkbox"/>	<input type="checkbox"/>	[]
Objects supported	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> Objects supported
Image scale:	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0.8
Convert tables to lists within lists.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> Convert tables to lists within lists.
Image quality resolution:	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Compromise between quality and size
Convert images into image links	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> Convert images into image links
Color is supported.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> Color is supported.

< Back Next > Cancel

Figure 16. Creating a new device preference profile

Note

All the options in this window are selectable. When selecting one, you can choose if it will be configurable or not on the Administration Console.

In the network profile, there are fewer options available for the image engine.

Create a New Preference Profile.

The following preferences can be included in this profile. You can include a preference if you want to change the default value for your device or network, if you would like the preference to be configurable from the Administration Console, or both.

Profile preference	Include?	Configurable?	Default value
Image types that are not supported	<input checked="" type="checkbox"/>	<input type="checkbox"/>	[]
Compress the size of HTML text files	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> Compress the size of HTML text files
File extensions that are not supported	<input checked="" type="checkbox"/>	<input type="checkbox"/>	[]
Remove images from HTML documents	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> Remove images from HTML documents
Image quality resolution:	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Compromise between quality and size ▼
Convert images into image links	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> Convert images into image links

< Back Next > Cancel

Figure 17. Creating a new network preference profile

2.6 Structure of Transcoding Publisher

The basic definition for the structure is to say that the IBM WebSphere Transcoding Publisher consists of a backbone and multiple transcoder plug-ins. This is used in the proxy model.

The backbone consists of:

- Rules engine to select actions for requests
- Configuration adapter for deployment options:
 - HTTP protocol engine
 - Servlet
 - Caching proxy access

- Tracing and logging mechanisms
- Access to configuration and profiles
- Preference aggregation mechanism

The backbone figures out which plug-in modules to run for each request, when to run them, and how to sequence them relative to each other. It also provides a means for plug-in modules to communicate with each other.

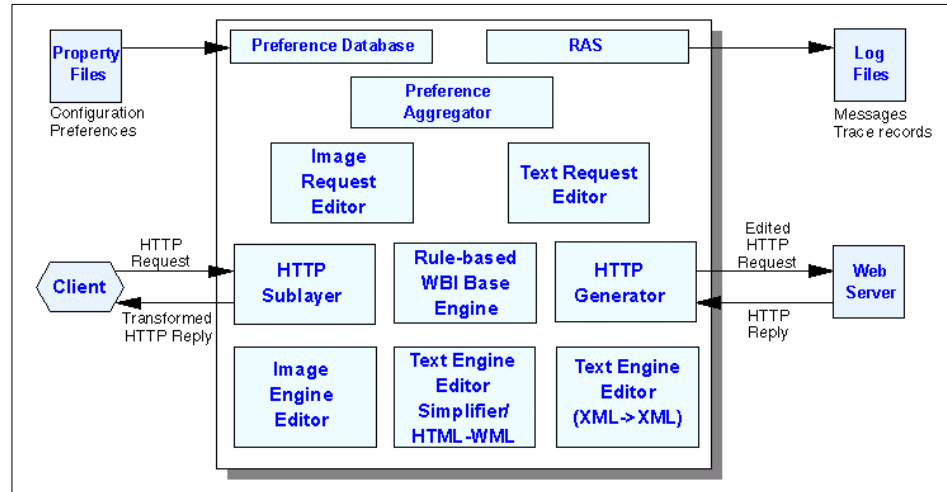


Figure 18. Backbone and Plug-ins in WTP Proxy model

A plug-in encapsulates a correlated set of transcoding components (MEGs). A MEG is one of **M**onitor, **E**ditor (Request or Document) or **G**enerator.

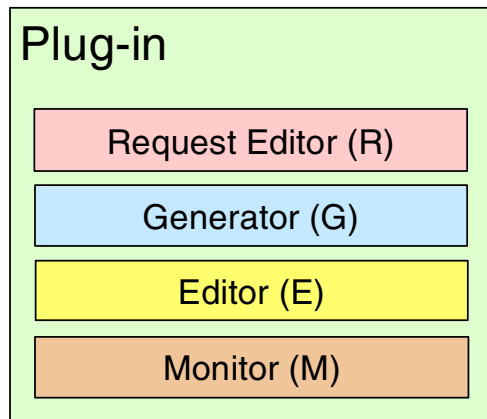


Figure 19. Plug-in architecture of the backbone

A plug-in consists of one or more MEGs. More information on MEGs and WBI (Web Intermediaries) can be found in the Web site:

<http://www.almaden.ibm.com/cs/wbi>

To find out which MEGs are invoked for a request can be determined using the RequestViewer tool (see 9.2, "The Request Viewer" on page 166) or with medium tracing enabled (see 13.3.2, "Tracing" on page 295).

2.6.1 Source identification

One of the major goals of WTP is to tailor the content sent to the user based on the user's environment, especially the device and connectivity limitations. There are two major steps needed to accomplish this:

1. Identify the characteristics, constraints, and preferences for each environment source. This step is called source identification.
2. Given multiple sources of preference information, provide the value of a given preference to use when requested by some transcoding logic. In WTP, the mechanism that provides this function is called the preference aggregator, because it "aggregates" multiple values into a single value.

2.6.1.1 Device identification

Identifying the device source type for an incoming HTTP request is based on the user-agent field in the HTTP header. For example, the User-Agent field for an English language version of the Netscape 4.7 browser running on a Windows NT system is "Mozilla/4.7 [en] (WinNT; U)".

Rather than requiring source identification on exact matches on the entire user-agent string, WTP supports source identification based on a portion of the user-agent string. For example, specifying the pattern “*Mozilla/4.*” would mean that any browser with a User-Agent string containing the String “Mozilla/4.” would match the pattern. Additionally, WTP supports device source identification via pattern expressions combined via logical operators. Thus, the expression:

```
(user-agent=*Mozilla/4.*) | (user-agent=*Mozilla/5.*)
```

means that any User-Agent String containing “Mozilla/4.” or “Mozilla/5.” would match the pattern expression.

2.6.1.2 Network source identification

When operating as a transcoding proxy, WTP supports the use of separate TCP ports to identify the type of network connection being used. That is, WTP can be configured to understand that port 8089 is a wireless network connection. Thus, if a client device’s browser is configured to connect to the WTP proxy via port 8089, a request from that device will then be assumed to be coming in via a wireless connection. Thus the wireless network profile will be used.

Note

When you run Transcoding Publisher as a WebSphere Application Server filter, it does not use network profiles.

2.6.2 Preference aggregation

For a given request, given that more than one preference source has a value for a given preference requested by a transcoding module, the preference aggregator must provide a single value. In most cases, for a given preference name, the value specified in one source profile takes precedence over the value in another source profile. For example, in a request from a Windows CE device using a wireless connection, the preference aggregator allows a precedence ordering to be specified to resolve any conflicts.

2.6.3 How a document is transcoded

When IBM WebSphere Transcoding Publisher receives a request from a user for a document, the flow follows these steps:

1. WTP determines the device type from the user-agent value in the document's HTTP header and locate the matching device preference profile. The user-agent value is defined when the device preference profile

is created. This information comes from the manufacturer of the device or you can use the Snoop tool to get this info from a new device and then create a new profile for that device. For details see 9.4, “The Snoop tool” on page 192.



Figure 20. Defining a user agent value

2. If network preferences are being used, WTP determines the network type from the port on which the request was received and access the matching preference profile.

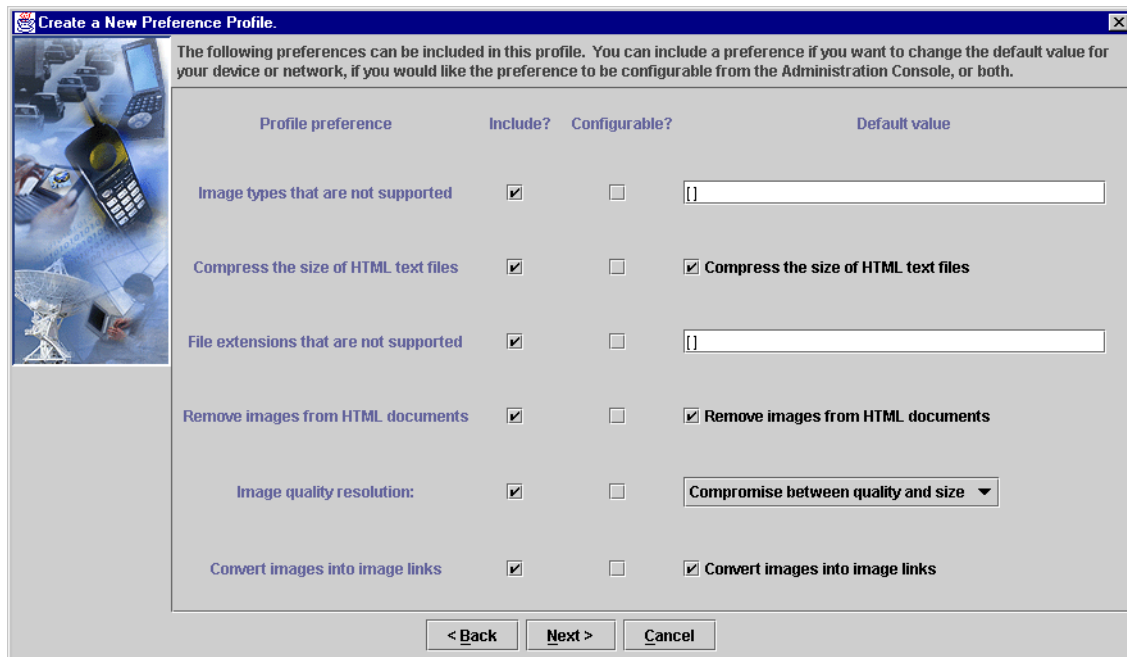


Figure 21. Creating a network preference profile

3. WTP resolves any preference conflicts in the preference aggregator.
4. WTP edits the request (request editor), modifying device restrictions that can be accommodated by transcoding. For example, for a device that does not support JPEG images, IBM WebSphere Transcoding Publisher can convert them to GIF images.
5. WTP retrieves the document from the firewall or Web server.
6. WTP schedules transcoders that match the preferences and document.
7. WTP executes the transcoders.
8. WTP returns the transcoded document to the requesting user.

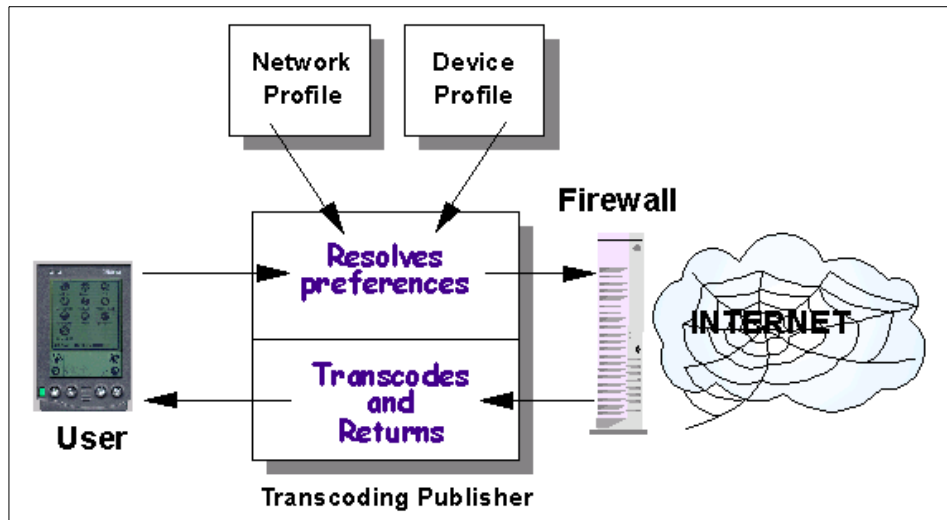


Figure 22. Basic flow of transcoded data

The steps are slightly different if you are using IBM WebSphere Transcoding Publisher as a network proxy with a cache server. This topic is covered in Chapter 6, “Running with a caching proxy” on page 73.

Chapter 3. Installation

In this chapter we describe the IBM WebSphere Transcoding Publisher Version 1.1 installation process on Microsoft Windows NT. This product can also be installed on IBM AIX, Microsoft Windows 2000, Sun Solaris, and Red Hat Linux. Instructions can be found in the Administration Guide and in the Readme file. English is the only language that is supported on Solaris and Linux. Any attempt to install or run this product under any other language on Solaris or Linux may produce unexpected results.

3.1 Planning for Transcoding Publisher

Before you install and configure Transcoding Publisher, there are several decisions you must make about how Transcoding Publisher will run. By reviewing these items now, you will be prepared to answer questions during installation and configuration.

3.1.1 Server configuration

The first decision you must make is how you will run Transcoding Publisher:

- As a proxy, also known as a network intermediary

In this configuration Transcoding Publisher is a single service that tailors content coming from many different Web servers. The proxy intercepts HTTP requests and responses as they flow between the user and Web server. This configuration does not tailor content that is encrypted between the user and the Web server.

If you run Transcoding Publisher as a network proxy, you can use it with or without a cache server. If you use a cache server in your network, Transcoding Publisher can use it to store and retrieve transcoded Web pages and intermediate results. This may enable Transcoding Publisher to avoid repeating the transcoding of frequently accessed pages.

If you run Transcoding Publisher as a network proxy, and your network uses a cache server or a firewall, you will need to supply the addresses and port numbers used by these servers when you configure Transcoding Publisher.

- As a filter running in IBM WebSphere Application Server

In this configuration, Transcoding Publisher tailors the content generated by a single Web server. When running as a filter, Transcoding Publisher can tailor content before it is encrypted and sent to a user. This configuration is not supported on Linux. When you run Transcoding

Publisher as a filter, you cannot use network preference profiles or WML deck fragmentation.

Transcoding Publisher works with WebSphere Application Server version 2.03. If you use WebSphere in your network, running Transcoding Publisher as a WebSphere filter enables it to tailor the output of other filters and servlets for your pervasive devices.

If you use encryption within the WebSphere Application Server, then you must use Transcoding Publisher as a filter so that it can modify content before it is encrypted. If you do not use encryption, then you could run Transcoding Publisher either as a filter or as a proxy. Running as a proxy, Transcoding Publisher can transcode non-encrypted documents from any version of WebSphere Application Server.

If you run Transcoding Publisher as a WebSphere filter, you must identify the servlets whose output you want Transcoding Publisher to transcode.

- **As JavaBeans**

The programs inside the Transcoding Publisher transcoding server that modify content are called *transcoders*. The transcoders are also provided as JavaBeans, which can be run independently of Transcoding Publisher. This provides a means for other server programs, such as servlets, independent content-providing programs, or JavaServer Pages (JSP), to invoke single transcoders directly. The JavaBean wrapper provides the transcoder with the same information about the system and request that it receives inside the Transcoding Publisher transcoding server, so that it operates the same way in both contexts.

The JavaBeans are always installed on AIX, Sun Solaris, and Linux; on Windows NT and Windows 2000 you have the option to install them or not.

3.1.2 Security

Neither the transcoding server nor the Administration Console provides a log-in mechanism. Each will be accessible to any user with permission to log on to the machine (either locally or by telnet) and execute commands in the Transcoding Publisher directories. If you want to limit access to Transcoding Publisher, consider installing it on a machine with limited access or in a file system with limited permissions.

3.1.3 Hardware and software prerequisites

If you run Transcoding Publisher as a filter in the WebSphere Application Server, Version 2.03, the prerequisites for Transcoding Publisher are the

same as those for WebSphere, except that you'll need another 30 MB of disk space.

To run Transcoding Publisher as a network proxy, use one of these configurations:

- **Windows NT:** Version 4 with Service Pack 5; Java Development Kit (JDK) 1.1.8 (available on the installation CD); Pentium(R) II system, 266 MHz or higher, with at least 128 MB of memory and 30 MB free disk space.
- **Windows 2000 Server:** Java Development Kit (JDK) 1.1.8 (available on the installation CD); Pentium II system, 266 MHz or higher, with at least 128 MB of memory (or the minimum required by the operating system) and 30 MB free disk space.
- **AIX:** Version 4.3.2 or higher; Java Development Kit (JDK) 1.1.8 (available on the installation CD); 166 MHz or higher system with at least 128 MB of memory.
- **Sun Solaris:** Version 7 or higher; Java Development Kit (JDK) 1.1.8; 166 MHz or higher system with at least 128 MB of memory.
- **Red Hat Linux:** Version 6.1; IBM Java Development Kit (JDK) 1.1.8 (available on the installation CD); Pentium II system, 266 MHz or higher, with at least 128 MB of memory and 30 MB free disk space.

3.2 Installing Transcoding Publisher

In this section we show you the installation for the Windows NT system. Before you begin the installation, be sure that your computer satisfies all the hardware and software prerequisites and that the current user has NT Administrator privilege.

3.2.1 Installation

Start the installation with the `setupwin` command from the IBM WebSphere Transcoding Publisher CD-ROM.



Figure 23. Transcoding Publisher installation

From this window you can view the documentation, explore the CD, visit the product support Web site, or install Transcoding Publisher.

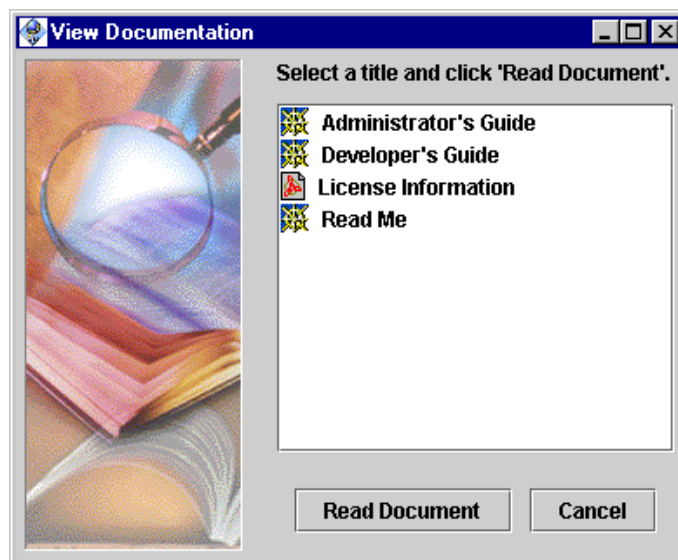


Figure 24. Transcoding Publisher online documentation

When you choose to install Transcoding Publisher, the Install Manager will search for the presence of the Java Development Kit (JDK) at 1.1.8 or higher, but less than 1.2. If the required JDK is not found, Install Manager will ask whether you want to install it from the installation CD. The JDK install is very simple and self-explanatory. Refer to the JDK documentation with any questions.

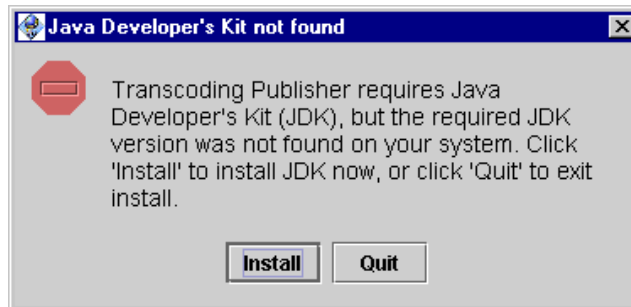


Figure 25. JDK not found warning

Installing this JDK will not interfere with any existing version that may be on the system, nor will it change any PATH or CLASSPATH settings. Transcoding Publisher's install sets its own environment variable.

Note

When you install Transcoding Publisher, the environment variable TP_JAVA_HOME is set to the location of the JDK. If you move the JDK after installing Transcoding Publisher, select **Settings>Control Panel>System>Environment** to update this variable to the new location.

After the JDK has been installed, you can return to installing Transcoding Publisher.

WebSphere Application Server is required if you want to run Transcoding Publisher as a servlet. Install Manager will determine whether the WebSphere Application Server is installed on your computer. If WebSphere is not installed, you will see a message saying that you will not be able to run Transcoding Publisher as a WebSphere filter on this machine, but if you want to run transcoding as a proxy rather than a servlet, then WebSphere Application Server is not required.

A warning is issued if WebSphere Application Server is not found, to make sure that you know that you will not be able to run transcoding in the servlet model.



Figure 26. WebSphere not found warning

If you want to run as a network proxy, you can ignore this message. If you want to run as a WebSphere filter, you must install WebSphere Application Server, Version 2.03. This does not actually affect the way that the install runs. The exact same set of files is laid down, regardless of whether or not WebSphere Application Server is present. The install has no knowledge of whether the product will run in proxy or servlet model - that's a configuration selection.

At this point, you will have to choose a setup language.



Figure 27. Choose Setup Language

When you click **OK**, InstallShield will display a standard welcome message, including copyright information.

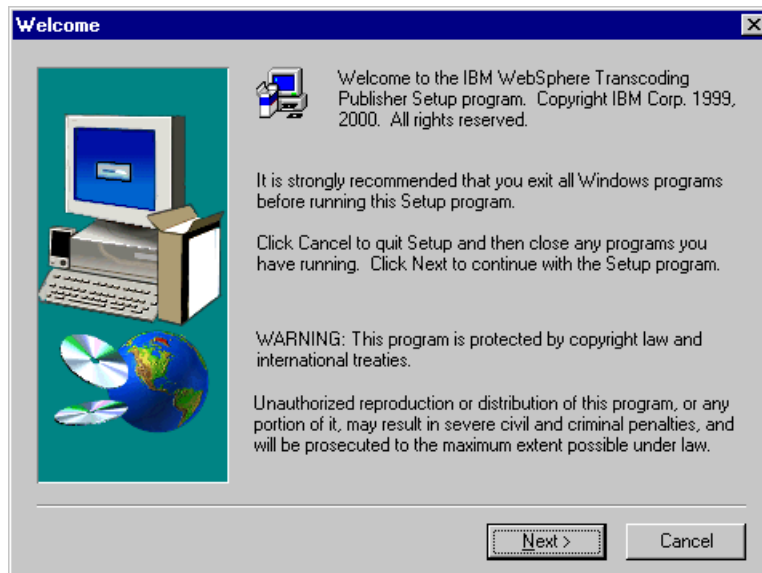


Figure 28. Welcome window

Click **Next** and InstallShield will prompt you to choose an installation directory. The default location is C:\Program Files\IBMTrans. You can change this to any desired directory.

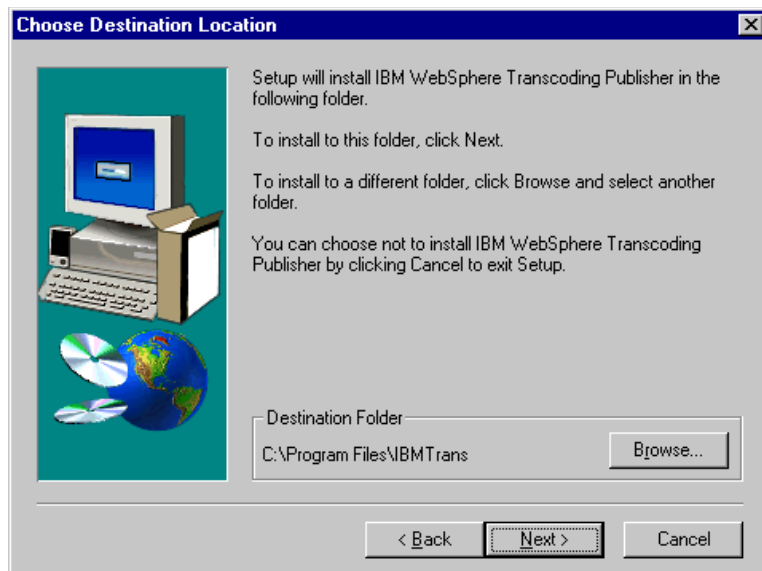


Figure 29. Choose target directory

By selecting the Next button, you can choose whether to install the base product, the documentation and the JavaBeans. You can install any combination of these components.

- Program Files includes all of the files necessary to run the product.
- Documentation includes both the documentation and help files, as well as the developer's toolkit executables.
- JavaBeans includes the JavaBean files that independent developers can incorporate into their own Java projects.

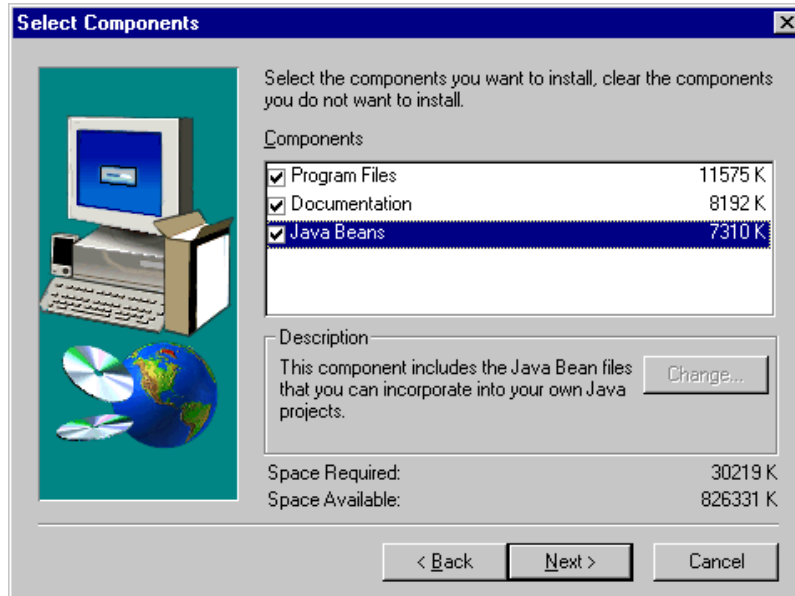


Figure 30. Select components to install

Note

You only need to install JavaBeans when you plan to invoke transcoding directly from your applications, for example servlets or JSPs. In this case, WAS can also be used but it is not required.

Shortcuts will be created under the Programs folder in the Start Menu. A Toolkit subfolder will contain additional shortcuts (only if the Documentation component with the Toolkit files was selected to be installed).

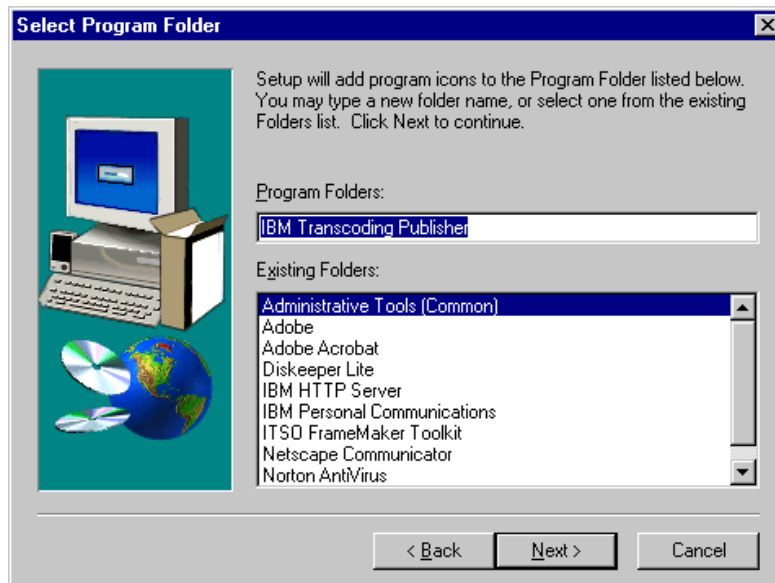


Figure 31. Adding program files to the program folder

Then, InstallShield will display a summary of all selections and ask you to change them or to proceed. This is the last chance to change anything before the copying of files.

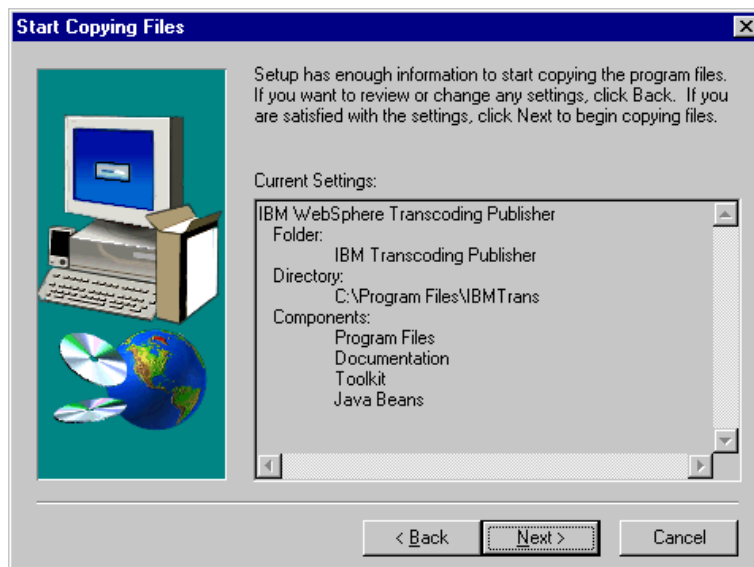


Figure 32. Start copying files

After you approve the installation options, InstallShield will copy all files to the target system. It registers the product as an NT Service - called IBM Transcoding Publisher Service.

When the installation is complete, InstallShield will display a completion message with two options:

- View the Readme file which will launch the default browser that's registered on the target system.
- Start the Server Setup Program which will initiate the configuration program that enables the user to configure Transcoding Publisher to run properly. This program is not part of the install - it's a completely separate configuration program.



Figure 33. Setup complete

In this redbook we show the Server Setup program in Chapter 4, “Configuration” on page 51. Therefore, we do not select the option to start the Server Setup program at this time (see Figure 33).

3.2.2 Logging

The install log file should be included with any defect reports to aid in problem solving, along with a description of exactly what the user was doing when the error occurred. In Windows NT, the file location is:

```
<product_dir>\log\tp_install.log
```

Chapter 4. Configuration

After you've installed Transcoding Publisher, it's time to configure the product before running it for the first time. The Server Setup wizard will configure the necessary options to implement the decisions you made in planning and installing.

When you run Transcoding Publisher as a network proxy, you can use network preference profiles to tell Transcoding Publisher how to transcode documents requested over different types of networks. Transcoding Publisher provides profiles for some network types, and you can add others. Transcoding Publisher determines the network type according to the port on which the request is received.

4.1 Server configuration

In this session, we describe the steps necessary to configure WTP as a network proxy without external cache server and without using a firewall server. This is the default configuration.

If not already configured, open the Server Configuration program by clicking **Start>Programs>IBM Transcoding Publisher>Server Setup**, as illustrated in Figure 34.



Figure 34. Server setup default location

The configuration process consists of a sequence of dialog boxes. After you start the configuration program, you will see a welcome window.



Figure 35. Transcoding Publisher server - setup window

Click the **Next** button, and you will be prompted to select the WTP model as follows:

- As a proxy in the network: Installed by itself, Transcoding Publisher acts as a network intermediary that can provide transcoding services for information provided by other Web servers.
- As a filter in the IBM WebSphere Application Server: This requires that IBM WebSphere Application Server is installed in your system. With WebSphere Application Server, WTP transcodes content produced by other servlets hosted by this application server. However, it does not transcode information provided by other Web servers.



Figure 36. Server setup configuration

Note

WebSphere Application Server is required if you want to run Transcoding Publisher as a servlet. If WebSphere is not installed, you will not be able to choose between proxy or filter. In this case, it assumes that you are installing as a proxy and you will not see this panel.

4.1.1 Configuring Transcoding Publisher as a network proxy

When you select the option to run Transcoding Publisher as a proxy, you must specify the other network servers with which it will need to interact. This may include a firewall server and a cache server.

Some networks use a cache server to store Web pages and other data, so if the same pages are requested frequently, they can be served from cache rather than repeatedly retrieved from external Web servers. If your network uses a cache server, and you want Transcoding Publisher to use it to store

transcoded versions of documents, you must configure Transcoding Publisher to use the cache server.

For information on how to configure WTP to use a cache server see Chapter 6, “Running with a caching proxy” on page 73.

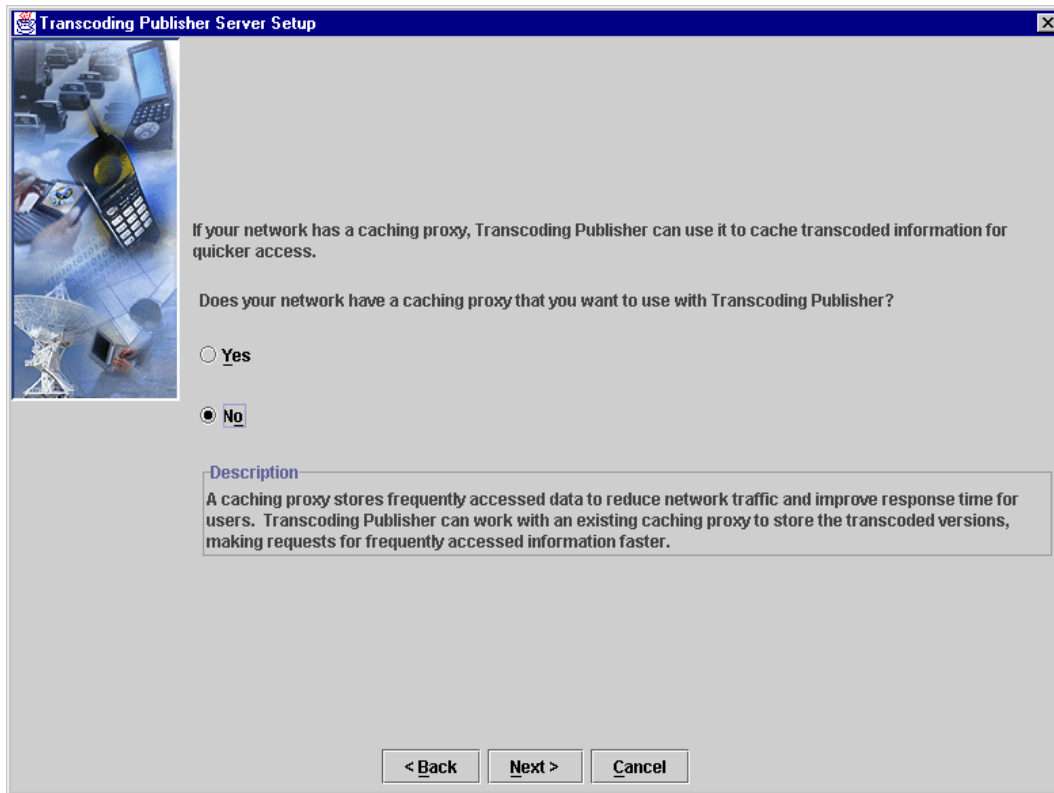


Figure 37. External cache option

Many companies use a firewall to protect their networks from intrusion by outside parties. The firewall might be a proxy server or a SOCKS server. All communications with addresses outside their networks are routed through the firewall.

If your network uses a firewall to communicate with Web servers outside your network, you must configure Transcoding Publisher to use the firewall. See 5.3.2, “Firewall configuration” on page 67 for details.



Figure 38. Choose firewall connection

You have to identify the type of firewall used in your network.

- None: your network does not use a firewall. Transcoding Publisher connects directly to the Internet with no intervening server.
- Proxy Server: Transcoding Publisher connects through a proxy server that is configured with a firewall to manage network traffic and to protect your network from outside intrusion.
- SOCKS Server: A SOCKS server is a proxy server that uses a special protocol, SOCKS, to forward requests. Transcoding Publisher connects through a SOCKS server that is configured with a firewall to manage network traffic and to protect your network from outside intrusion (It supports Versions 4 and 5 SOCKS servers).

In this scenario, we will not use external cache server or a firewall.

Now we have finished the configuration with the necessary options to enable Transcoding Publisher to run successfully. Specific configuration of this

product using other scenarios will be covered in more detail in the other chapters.

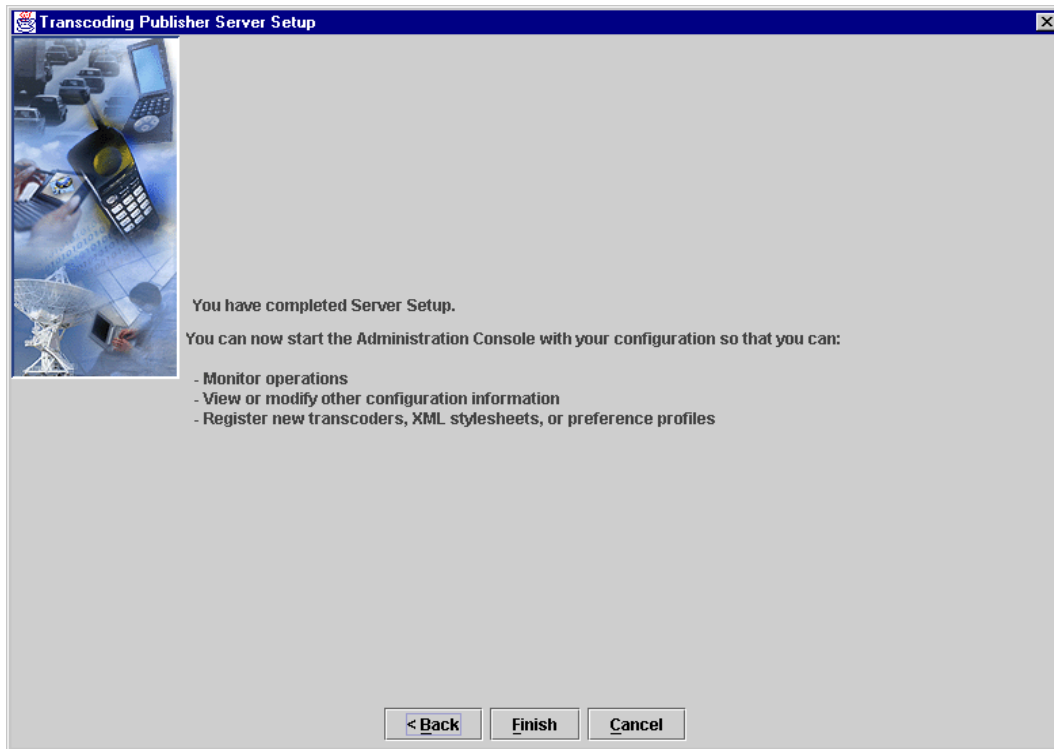


Figure 39. Server setup complete

Whenever you create a new configuration or change the existing configuration, it requires you to restart the IBM Transcoding Publisher service so that the changes will take effect. The next window is just a reminder.

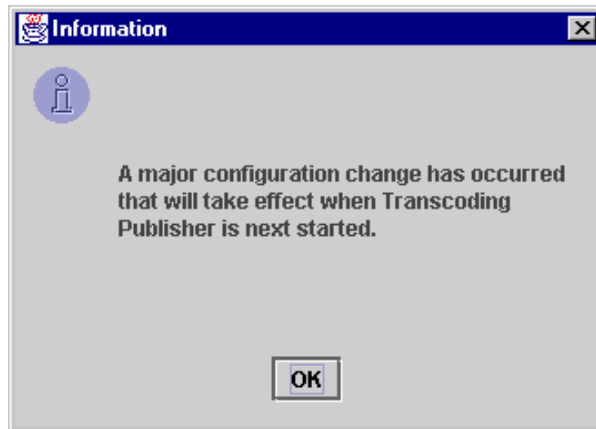


Figure 40. Change configuration window

4.2 Client configuration

If you add Transcoding Publisher as a filter in a network that already uses WebSphere Application Server, you do not need to change the configuration of your client devices.

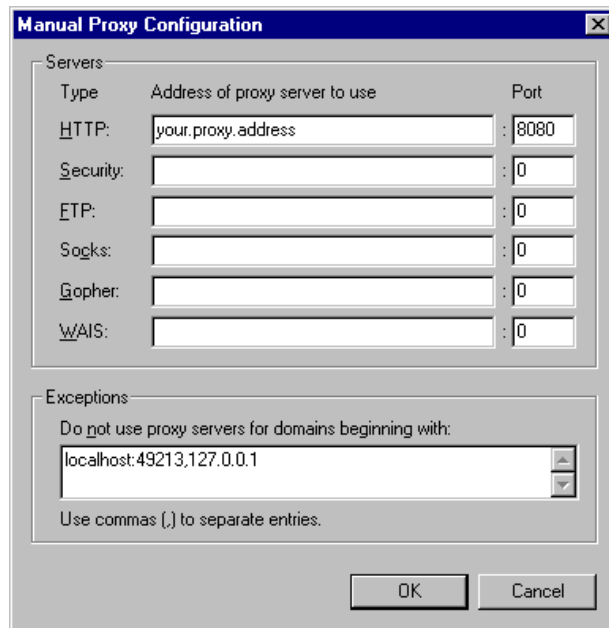
If you use Transcoding Publisher as a network proxy, you must configure your client devices to use your Transcoding Publisher server as an HTTP proxy.

Your clients may already be configured to use your firewall as a proxy server; in that case, you can simply replace the firewall's address with the address of the machine where you installed Transcoding Publisher. Instead of using numbered IP addresses you can use the fully qualified host name for the server where WTP is running.

If your clients are configured to use the firewall as a SOCKS server, you don't need to change that. You only have to include the address of the Transcoding Publisher proxy in the HTTP proxy address. In this case, you will use WTP to the HTTP proxy server and the SOCKS server for other types of requests.

Note

If you want to use network preferences profiles, be sure to configure each client to access Transcoding Publisher on the port corresponding to the type of network it uses.



Manual Proxy Configuration

Servers

Type	Address of proxy server to use	Port
HTTP:	your.proxy.address	8080
Security:		0
FTP:		0
Socks:		0
Gopher:		0
Wais:		0

Exceptions

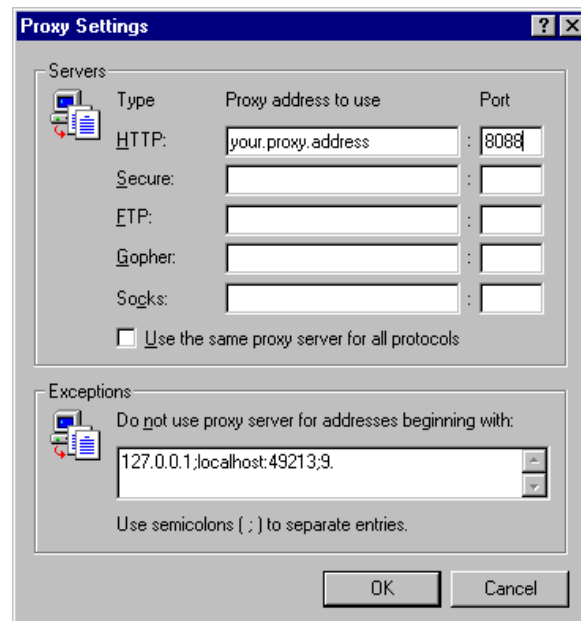
Do not use proxy servers for domains beginning with:

localhost:49213,127.0.0.1

Use commas (,) to separate entries.

OK Cancel

Figure 41. Netscape manual proxy configuration



Proxy Settings

Servers

Type	Proxy address to use	Port
HTTP:	your.proxy.address	8080
Secure:		
FTP:		
Gopher:		
Socks:		

☐ Use the same proxy server for all protocols

Exceptions

Do not use proxy server for addresses beginning with:

127.0.0.1;localhost:49213;9

Use semicolons (;) to separate entries.

OK Cancel

Figure 42. Internet Explorer manual proxy configuration



Figure 43. Proxy settings for Nokia WAP toolkit

When using the Nokia Phone Simulator Toolkit you can configure the preferences to use your WebSphere Transcoding Publisher proxy.

Each network profile has a port associated with it. For the default network profile the port number is 8088. You can change it after installation. See Chapter 5, “Running as a stand-alone network proxy” on page 63 for more details on port configuration.

4.3 Changing your configuration

If, after you have configured and run Transcoding Publisher in one way, you want to change to a different configuration, you can make this change by invoking the Server Setup wizard (see 4.1, “Server configuration” on page 51). You can start the wizard from the Administration Console by selecting **File->Settings->Server setup**. Or you can start the wizard from the IBM Transcoding Publisher program group in the Windows Start menu.

This wizard is needed only to switch between running as a proxy and running as a WebSphere filter, or to add or delete the ability to work with a cache server. All other changes can be made from within the Administration Console.

If you change your configuration using the server setup wizard, you must restart the transcoding server. If you have changed Transcoding Publisher to a WebSphere filter, complete the configuration and restart WebSphere.

If your reconfigured Transcoding Publisher will run as a network proxy, and the transcoding server is running, the method to restart it depends on your operating system. On Windows NT and Windows 2000, use the Services utility on the Control Panel to stop and restart the IBM Transcoding Publisher service.

Note: When you restart the WebSphere Application Server (WAS), you should actually do this before starting the WAS Administration tool. The correct procedure is:

1. Stop WAS before running the server setup wizard. Since the wizard is invoked at the completion of the Transcoding Publisher installation process, you might want to stop WAS before installing WTP.
2. Run the server setup wizard.
3. Restart WAS to read the changes made by the server setup wizard, then start the WAS Administration tool.

4.4 Configuring Transcoding Publisher to start automatically

The following information applies only when Transcoding Publisher is configured as a network proxy. When it is configured as a WebSphere filter, the WebSphere Application Server will start and stop Transcoding Publisher.

On Windows NT, the Transcoding Publisher transcoding server is installed as a Windows NT service. It is set to be started manually. To have Transcoding Publisher start automatically when your system is started, use the Services utility on the Control Panel. Select IBM Transcoding Publisher in the list of services, then click **Startup**. Change the startup type to automatic. Transcoding Publisher will be started automatically when the machine is started.

4.5 Preference profiles

For information on how you configure network and device profiles see 2.5.1, "Preference profiles" on page 20.

4.6 Transcoders

For information about transcoders see 2.5.3, “Transcoders” on page 27.

4.7 XML stylesheet selectors

For information about XML stylesheet selectors see 2.5.2, “XML stylesheets” on page 25.

4.8 Administration Console

For information about the IBM WebSphere Transcoding Publisher Administration Console see 12.2, “Using the Administration Console” on page 261.

Chapter 5. Running as a stand-alone network proxy

In this chapter we show you the scenario with IBM WebSphere Transcoding Publisher using the stand-alone proxy model. This is the simplest option and the easiest to get started. It is also the one that developers will probably use to develop and test new transcoders.

5.1 Overview

In a typical Web environment, information is simply sent from a server to a browser for display and interaction. However, there are many ways that adding an intermediary between the browser and the server can improve the system. For example, an intermediary can keep track of the information the user has viewed in order to make it easier to find information again. Or, an intermediary may enhance the information that the user sees by adding annotations and personalization beyond what the server was designed to do. Intermediaries turn the network into a “smart pipe” with applications that can enhance the information on the Web.

With this option, WebSphere Transcoding Publisher sits between the client and any Web server that can be reached by the client. It can have a firewall between it and the Web server.

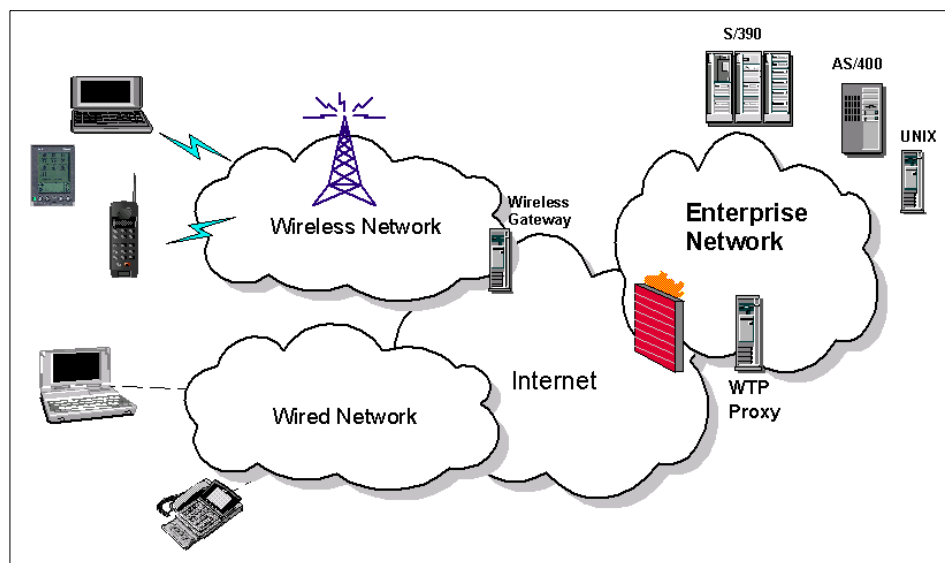


Figure 44. Basic proxy model

In the emerging pervasive market place you have a wide variety of devices attempting to access data on the Internet and within enterprise systems. It should be noted that there is a complex underlying structure of wired and wireless gateways at play in this environment. However above the base level of getting connected is the flow of HTTP requests. In the basic transcoding model these flow from the client device to a proxy.

The transcoding framework also allows industry-standard servlets to be incorporated as transcoders. Servlets written for other environments can be easily ported to run in the proxy, and new plug-ins written to the Java Servlet API will be usable in the many Web servers that support servlets. The transcoding proxy supports Version 2.1 of the Sun Java Servlet API. Servlets written to this API should run with few or no changes in the transcoding framework.

5.2 Basic flow for Transcoding Publisher acting as a proxy

The basic flow for Transcoding Publisher acting as a network intermediary (transcoding proxy) is shown below.

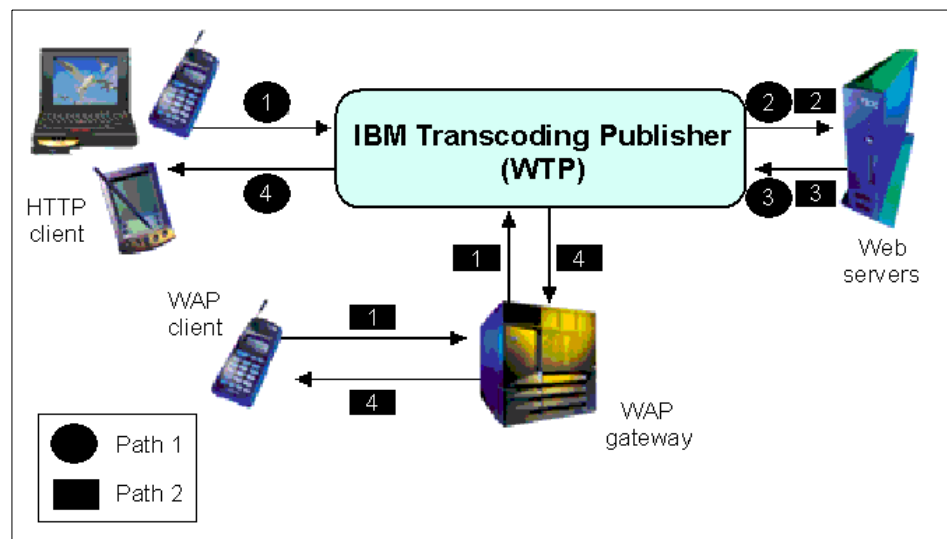


Figure 45. WTP basic flow acting as a proxy (Path 1 and Path2)

In the diagram, the arrows show the flow of information to and from the transcoding proxy. The flows are numbered to show different paths through the transcoding proxy.

Path 1 shows the transcoding proxy acting on behalf of an HTTP-based client browser that expects HTML or XML pages in return.

1. The client browser sends an HTTP request to the transcoding proxy. The request can be for any Web documents, including HTML pages, XML pages, or GIF or JPEG images.
2. The transcoding proxy can edit the request to modify the URL or to change values in the header fields. It then sends the HTTP request to the Web server to acquire the requested object. The proxy also saves information from the request that identifies the device making the request.
3. The Web server returns the page requested by the transcoding proxy on behalf of the client.
4. The transcoding proxy uses information from the request and response along with rules in the Preference Aggregator to identify the device, user, and network profiles needed to control the way the document is transcoded. When particular transcoding modules running in the transcoding proxy request the values of particular preferences or constraints, the Preference Aggregator uses the profile identification information in evaluating its rules to help determine the correct value to return. Using these preference and constraint values, the transcoding proxy edits the response received from the Web server, tailoring it for the client device before returning it to the client in an HTTP reply.

Path 2 shows the transcoding proxy acting on behalf of a Wireless Application Protocol (WAP) client, such as a smart phone.

5. The wireless client sends a WAP request that is converted to an HTTP request by the WAP gateway. The WAP gateway forwards the HTTP request to the transcoding proxy.
6. The transcoding proxy sends the HTTP request to the Web server after any necessary editing.
7. The HTTP reply comes back from the Web server.
8. When the transcoding proxy matches information from the request against device and user profiles, it discovers that the device requires Wireless Markup Language (WML) output. For this example, assume the original document was an XML document. The transcoding proxy uses the preference information to select the correct stylesheet to convert the document to its WML form. Then it returns the response to the WAP gateway, which converts the text-based WAP Markup Language documents to the compressed wireless-ready form before sending it to the client smart phone.

5.3 Configuration

If you have already installed and configured as shown in Chapter 4, “Configuration” on page 51, you have already configured your Transcoding Publisher as a network stand-alone proxy. This section goes into the previous configuration in more detail.

5.3.1 Port mapping

There are three network preference profiles provided when you finish WTP as a proxy configuration. You can change the port associated with each of these profiles manually or you can use the Administration Console.

Even if you are not adding any network preference profiles, you should review the default port settings for the network preference profiles provided with Transcoding Publisher to ensure that they do not conflict with existing port settings on your system. If there is a conflict, you can use the Proxy Port Settings panel to change the port settings for the networks, or, if there is a network type you do not use, you can disable that network preference profile.

To open the Proxy Settings, just click **File -> Settings -> Proxy Port** on the Administration Console.

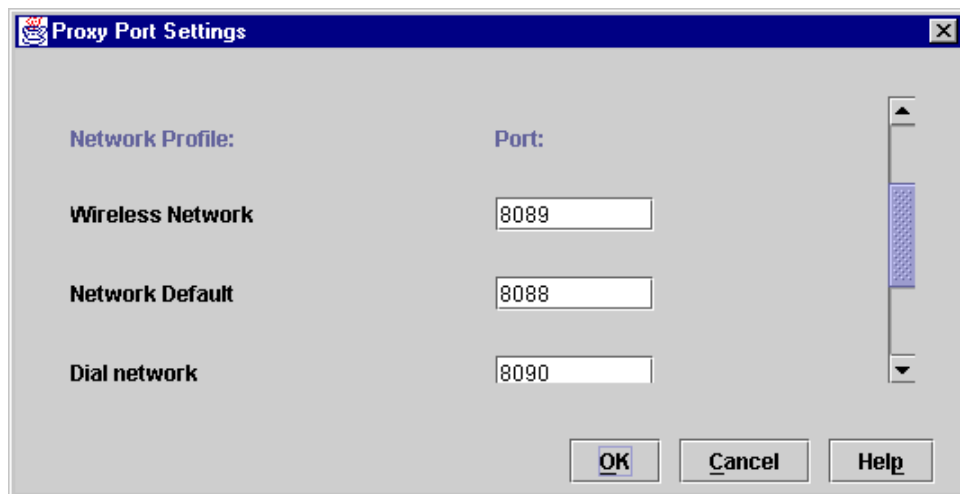


Figure 46. WebSphere Transcoding Publisher port mapping

This window will show only the enabled profiles. If you disable one of the profiles on the Administration Console, you will not be able to see the port associated with it, but it will keep the configuration for that profile, including

the port number. It will also show you the port for any enabled configuration that you have created and registered. To read more about creating and registering new preference profiles see 9.3, “Creating preference profiles” on page 174 and Chapter 12, “Administration” on page 261.

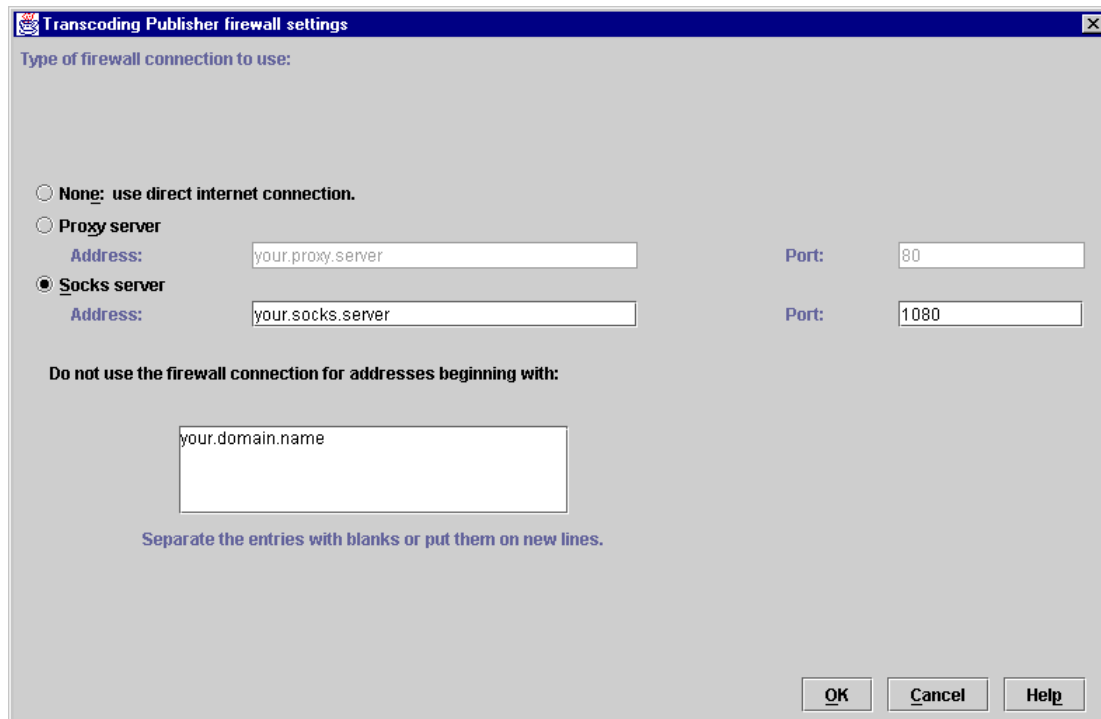
Note

Do not delete a profile from the Administration Console. If you don't want to use it just disable it. Otherwise, the file with all the information for that profile will be physically deleted from your computer.

5.3.2 Firewall configuration

If you have a firewall configured on your LAN, you must select the type of connection by which to access a Web server on the internet. It can be a socks or a proxy configuration. Please contact your network administrator to help you obtain the actual configuration.

To change your firewall configuration, on the Administration Console, click **File -> Settings -> Firewall**.



The image shows a Windows-style dialog box titled "Transcoding Publisher firewall settings". It has a standard title bar with a close button. The main area is light gray. At the top, it says "Type of firewall connection to use:". Below this are three radio button options: "None: use direct internet connection.", "Proxy server", and "Socks server". The "Socks server" option is selected. To the right of the "Proxy server" option are two text input fields: "Address:" with the placeholder "your.proxy.server" and "Port:" with the value "80". To the right of the "Socks server" option are two text input fields: "Address:" with the placeholder "your.socks.server" and "Port:" with the value "1080". Below these options is a section titled "Do not use the firewall connection for addresses beginning with:". It contains a large text input field with the placeholder "your.domain.name". Below this field is a note: "Separate the entries with blanks or put them on new lines." At the bottom right of the dialog are three buttons: "OK", "Cancel", and "Help".

Figure 47. Firewall settings

Here you can specify the service that is being used on the firewall (proxy or SOCKS) with the associated port. You can also include the addresses for which the firewall will not be used.

5.4 Encryption support

Most of the sites uses encryption to keep information within that page secure. Actually, just a small part of these pages are encrypted, normally when dealing with personal information, bank information, or when buying or selling online. The other pages on the same site probably will not use encryption.

Transcoding Publisher running as a proxy server does not make any attempt to decrypt the information that the client and the Web server exchange during an encrypted connection. It establishes a connection to the destination Web server and passes the request to it without looking at the data.

Transcoding Publisher supports Secure Sockets Layer (SSL) connections. SSL secure connections involve encryption and decryption processes and

are established directly between the client browser and the destination Web server.

Running as a proxy and using encryption between the Web server and the browser, IBM Transcoding Publisher will not transcode any information. The information is already encrypted, so it cannot effectively modify encrypted data.

If you want to transcode this type of information you will have to transcode it before it is encrypted. To do this, you need to run Transcoding Publisher as servlet filters and transcode it at the source. See for example, Chapter 7, “Transcoding with WAS filters” on page 93.

Otherwise, if you want to transcode only the pages that are not encrypted and want to see the encrypted pages without transcoding, you only need to configure WTP as a proxy and your browser to support security on the manual proxy configuration. No encrypted pages will be transcoded and the proxy will run as a regular network proxy. That is, it will do a passthrough and will not handle (transcode) the client requests and server responses at all.

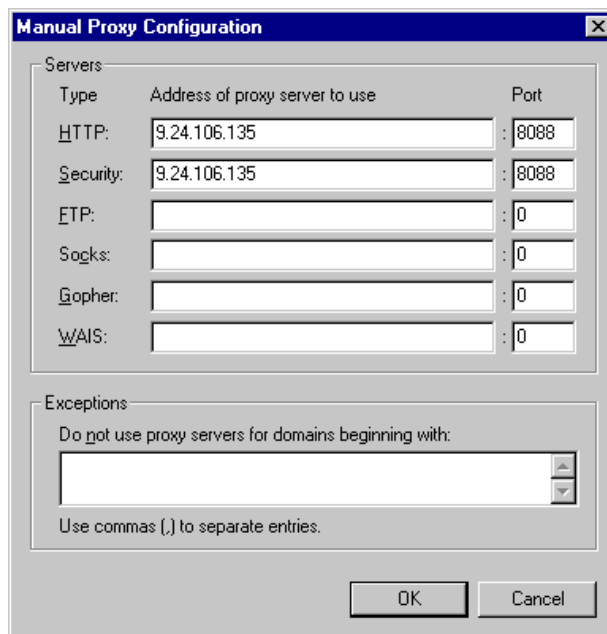


Figure 48. Netscape manual proxy configuration with security

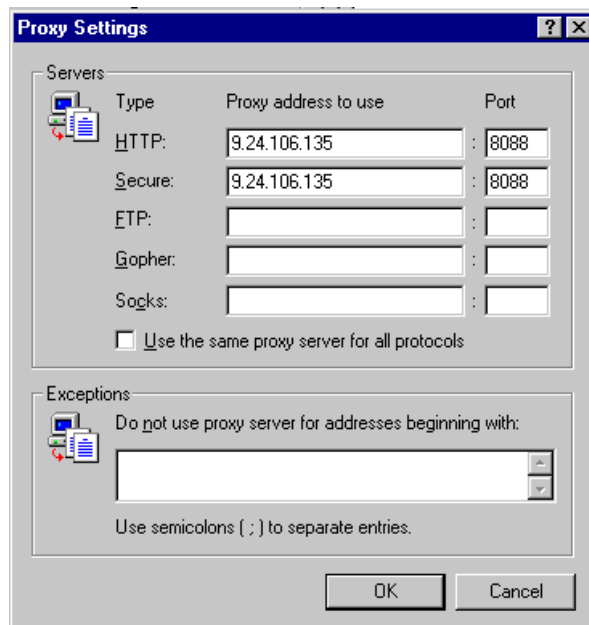


Figure 49. Internet Explorer manual proxy configuration with security

5.5 Request Viewer

You can use this tool to make sure that Transcoding Publisher is running as a proxy and the information is being transcoded.

This tool is part of the IBM WebSphere Transcoding Publisher (WTP) Toolkit. To start the Request Viewer, you need to click **Start -> Programs -> IBM Transcoding Publisher -> Toolkit -> Request Viewer**.

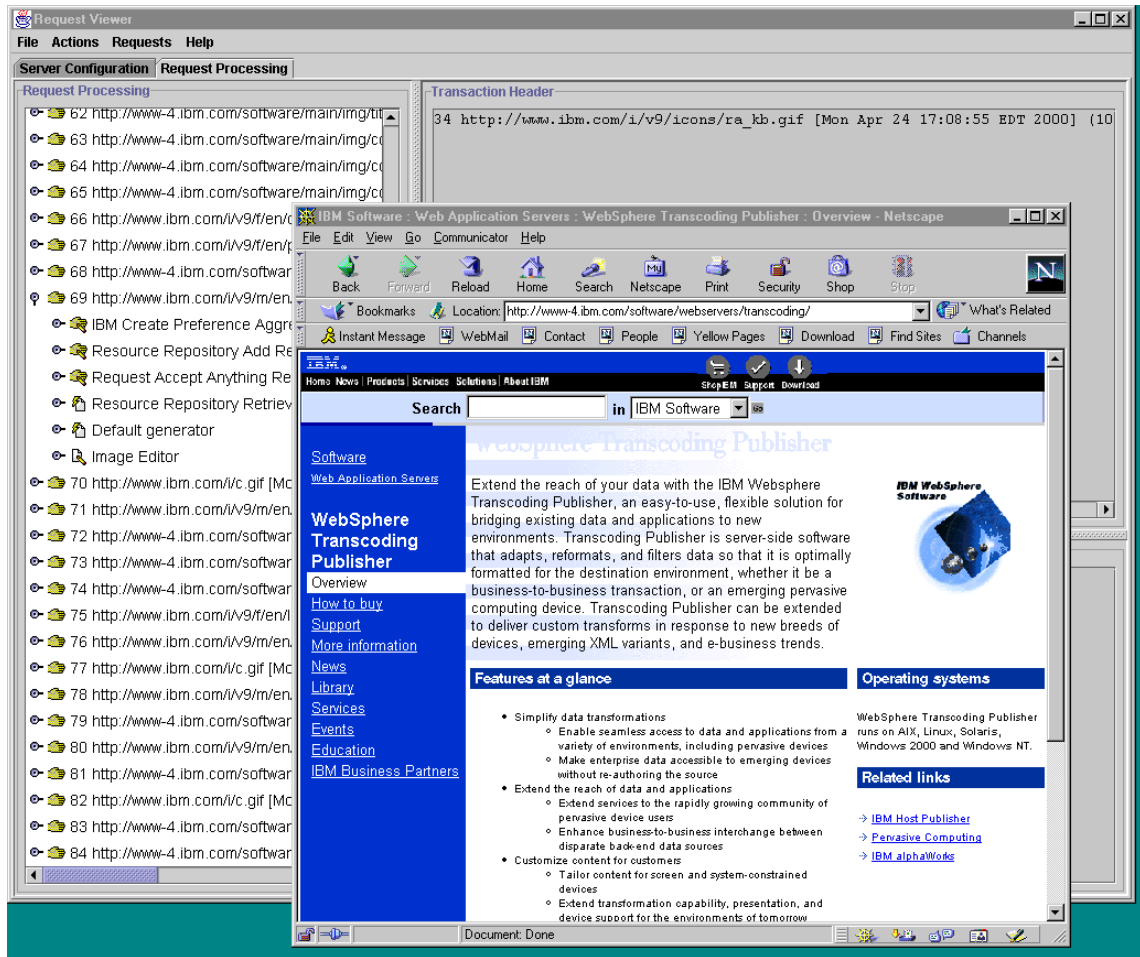


Figure 50. Request Viewer - request processing

Note

If you want to use the Request Viewer, the IBM Transcoding Publisher Service must be stopped. When you open this tool, it will start the same service but “internally”, so when you close Request Viewer, this internal service will be closed too. Remember to start the service when you stop the Request Viewer.

This tool is very useful, but you have to be careful to use tracing and messaging only when needed because it will make your proxy run slowly.

Note

The Request Viewer tool is intended to be used for development only. It is not designed and should not be used in a production environment.

Another restriction in this release is that you cannot save or copy the Request Viewer information using the options in the menu bar. Therefore, to copy and paste, select the text and use the Ctrl-C/Ctrl-V keys. For more information about this tool, see 9.2, “The Request Viewer” on page 166.

5.6 Troubleshooting

Make sure you have connectivity to the WTP proxy. If you cannot ping it from your client machine, then your browser will not be able to find it either.

If running on a local machine check the settings for bypassing local addresses. View the trace from the Administration Console to make sure you are reaching the WTP Proxy if you don't think your requests are being transcoded.

For more information about troubleshooting problems see Chapter 13, “Problem determination” on page 287.

Chapter 6. Running with a caching proxy

Some networks use a cache server to store Web pages and other data, so that if the same pages are requested frequently, they can be served from the cache rather than repeatedly retrieved from external Web servers. In this chapter we document how you implement IBM WebSphere Transcoding Publisher (WTP) to work with a cache proxy server such as WebSphere Traffic Express (WTE).

6.1 Overview

If your network uses a cache server, and you want Transcoding Publisher to use it to store transcoded versions of documents, you must configure Transcoding Publisher (WTP) to use the cache server.

The external cache is an HTTP proxy such as IBM Web Traffic Express (WTE), IBM's cache in the WebSphere Performance Pack, Squid, the open source cache, Microsoft Proxy, Netscape Proxy or Wingate.

WebSphere Transcoding Publisher has been successfully tested with WTE and Squid. Although not formally tested, it should also function with any other product compliant with the HTTP specification. For example, see 6.4.5, "Sample configuration for WinGate" on page 83. Also, there has not been any development effort to make this caching extension work with the WebSphere Application Server environment where WTP is configured as a MIME-type filter servlet.

6.2 Relationship to external cache

HTTP responses from Web servers may contain an explicit expiration which the external cache will obey (the HTTP header is *Expires*: followed by an HTTP compliant date). If this header is not provided, most servers will provide a last modified header for the resource (this header is *Last-Modified*: followed by an HTTP compliant date). The external cache will use the last-modified date and may cache the document for a period of time. Caching of documents without the expiration header is implementation specific (WTE may calculate the expiration based on the last-modified header using a different process than Squid). Most external caches that support caching based on last-modified headers will periodically verify the cached copy using "get if-modified-since" requests.

A server may also flag a resource as non-cacheable by adding either a *Pragma: No-cache* or *Cache-control: No-cache* header. It is also possible for a client to explicitly request a fresh copy (a full reload) using the *Pragma: No-cache* header on its request. This happens, for example, when you click the Reload button.

Caching transcoded results or variants is accomplished by splitting the WTP transcoding operation into two stages handled by the client and the transcoding subsystem:

- Preference aggregation -- the request is qualified with information which determines the particular variant or transcoded result.
- Transcoding of resource based on aggregated preferences.

Preference aggregation as referred to here is the process where the client's device, network type, and any other information that might cause the transcoders to produce a different variant (or permutation) of the resource, are determined.

Transcoding as referred to here is the process where the MEGs related to modifying the request, generating the original resource and all of the document (or resource) editing (or transcoding) occurs.

6.3 Understanding caching

When you use a cache server with Transcoding Publisher, Transcoding Publisher stores transcoded pages in the cache. Whenever a request is received, Transcoding Publisher asks the cache for the specific document. Then, the cache component makes the decision to either return the cached document to the user or to ask the WTP transcoding subsystem for a new copy of the document. In the latter case, Transcoding Publisher will retrieve it, transcode it, and store it in the cache before returning it to the user.

The HTTP 1.1 protocol enables Web sites to specify that a page cannot be cached. For example, some main site pages, secure and session-based pages cannot be cached. If the original document cannot be cached, transcoded versions of it cannot be cached either.

The flow of data to retrieve a page that has not been cached follows these steps (see Figure 51 on page 75):

1. Transcoding Publisher receives a request from a client for
`http://www.ibm.com`

2. Transcoding Publisher determines the device and network and accesses the appropriate preference profiles.
3. Transcoding Publisher resolves any preference conflicts.
4. Transcoding Publisher requests a suitably transcoded version of `www.ibm.com` from the cache.
5. The cache doesn't have a copy of `www.ibm.com`, so it asks the transcoding sublayer of Transcoding Publisher for it.
6. Transcoding Publisher sees a new request for the transcoded version of `www.ibm.com`, and so it asks the firewall for `www.ibm.com`.
7. The firewall asks `www.ibm.com` for the original resource and sends it back to Transcoding Publisher.
8. Transcoding Publisher runs the transcoders appropriate for this document and this user.
9. Transcoding Publisher sends the transcoded document to the cache, in response to the request in step 5.
10. The cache keeps a copy of the resource (according to HTTP caching rules on expiration, permission to cache, etc.) and sends the transcoded version back to answer Transcoding Publisher's original request in step 4.
11. Transcoding Publisher returns the transcoded resource to the client.

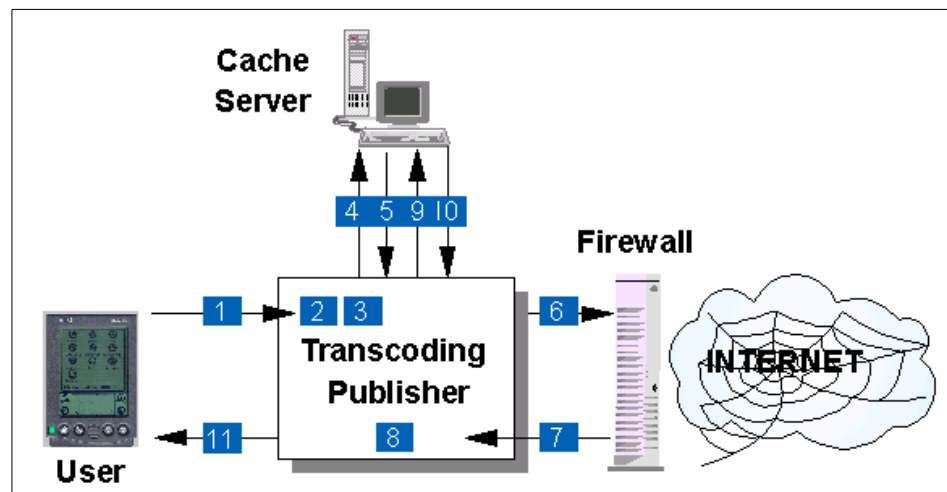


Figure 51. Understanding caching steps - first request

The next time a user requests the same page and transcoded in the same way, the flow will be much simpler:

1. Transcoding Publisher receives a request from a client for example, for `http://www.ibm.com`.
2. Transcoding Publisher requests a transcoded version of `www.ibm.com` from the cache.
3. The cache returns the transcoded version of `www.ibm.com` that it has stored.
4. Transcoding Publisher returns the transcoded page to the user.

When Transcoding Publisher requests a transcoded document from the cache, all these conditions must be true:

- The requested URL must match.
- The device type in the new request, as determined from the user agent field, must match the device in the original request.
- The network type in the new request, as determined from the port on which the request was received, must match the network type in the original request. If you are not using network preferences, and all requests are being received on the default port, then this condition is satisfied.
- The cached transcoded resource has not expired, either through an explicit expiration on the resource from the original server, or because the cache has been configured to expire documents of a certain age. For documents with no expiration time specified, the cache will tell Transcoding Publisher to check the original server to see whether the document has been modified.
- The cached transcoded resource has not been removed from the cache because the cache is reaching capacity.
- The request from the device must not be for a refreshed copy of the document. The cache will not return transcoded output if the browser specifies a forced reload.

If all these conditions are met, the cache will return the requested resource and Transcoding Publisher will send it to the user.

6.4 Configuration

When you run Transcoding Publisher as a network proxy with caching, you must specify the other network servers with which it will need to interact. These may include a firewall server and a cache server. If you want to configure the firewall settings see 5.3.2, “Firewall configuration” on page 67.

In the Server Setup wizard, you have to select the option to use a caching proxy.

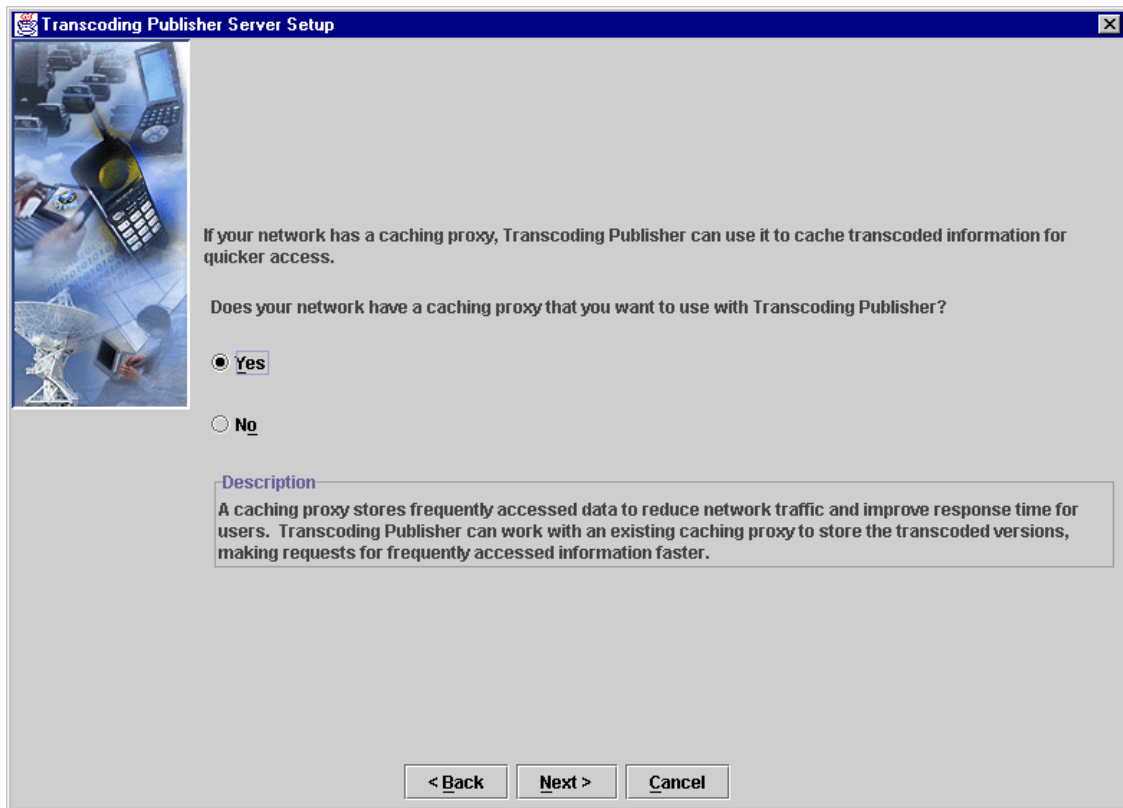
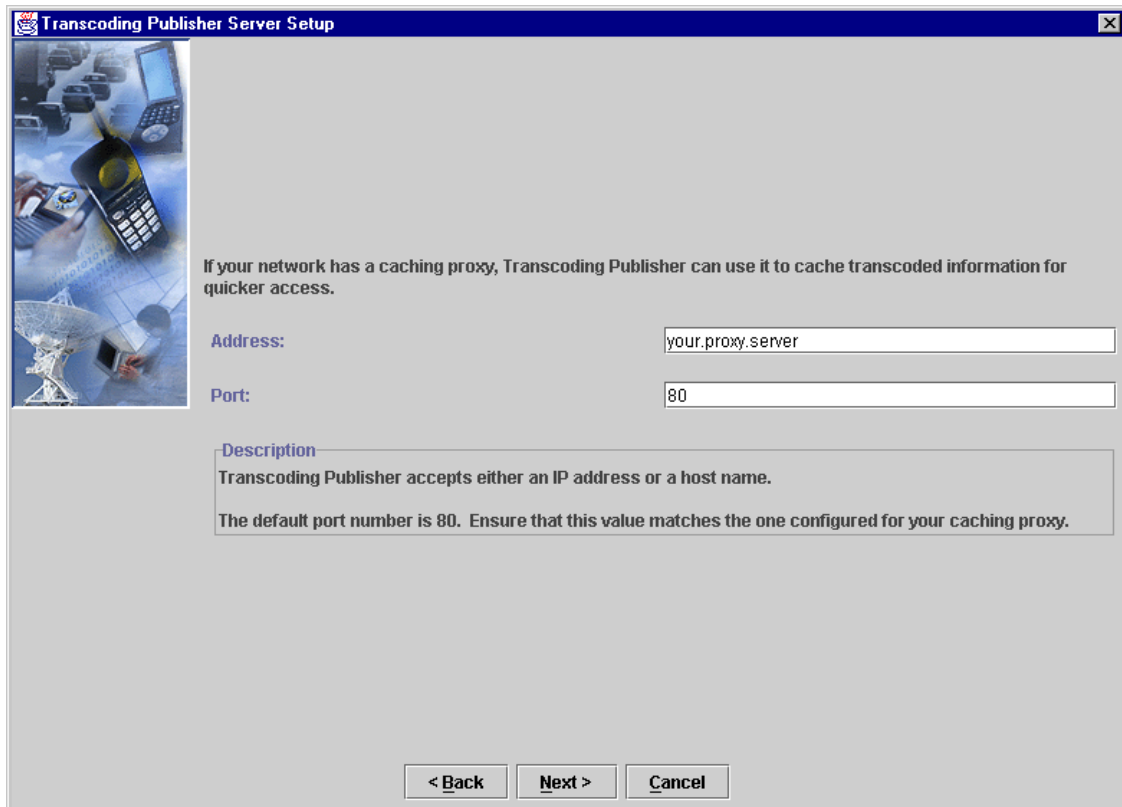


Figure 52. Server setup - proxy configuration

Click **Yes**, and then you have to specify the address and listening port of the cache server. You can specify the address in dotted decimal format or as a hostname. If you enter a hostname, Transcoding Publisher will attempt to validate the hostname through your Domain Name Server (DNS). This may cause a slight delay in processing the address.



The screenshot shows a Windows-style dialog box titled "Transcoding Publisher Server Setup". On the left is a vertical image strip containing icons of a car, a mobile phone, a PDA, and a satellite dish. The main area of the dialog has a light gray background. It contains the following elements:

- A text block: "If your network has a caching proxy, Transcoding Publisher can use it to cache transcoded information for quicker access."
- An "Address:" label followed by a text input field containing "your.proxy.server".
- A "Port:" label followed by a text input field containing "80".
- A "Description" section with a thin border containing the text: "Transcoding Publisher accepts either an IP address or a host name. The default port number is 80. Ensure that this value matches the one configured for your caching proxy."
- At the bottom, three buttons: "< Back", "Next >", and "Cancel".

Figure 53. Server setup - cache server configuration

This is the minimum configuration required for Transcoding Publisher to work with a cache server. To specify more settings, open the Administration Console and click **File -> Settings -> Cache**.

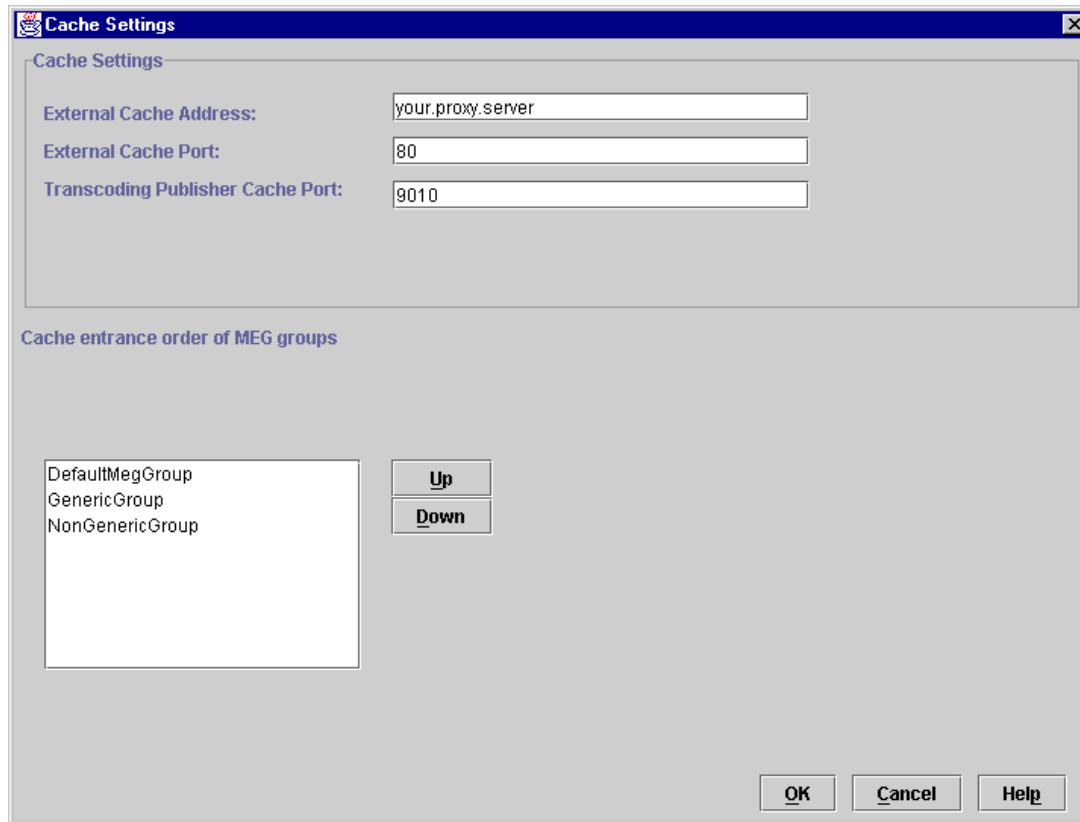


Figure 54. Administration Console - cache settings

Through the AdminConsole it is also possible to set the port that WTP will reserve to receive requests from the cache (the default is 9010).

The group order box in this dialog is discussed later in 6.4.3, “MEG groups and how they are used” on page 81.

6.4.1 Considerations for configuring the external cache

When the external cache option is used, the external cache must be configured to allow caching of documents in the domain where Transcoding Publisher is running. This means, if Transcoding Publisher is running on machine `transcode.ibm.com`, the cache server must be configured to allow caching in the `*.ibm.com` domain.

If the external cache resides on the same machine as Transcoding Publisher (that is, if the cache and Transcoding Publisher are both installed on

transcode.ibm.com), then the cache must be configured to allow caching of *.ibm.com, transcode.ibm.com, and localhost and 127.0.0.1 (the reserved IP address that represents the local machine).

If your cache uses exclusion to determine whether a document can be cached, remove any exclusions for the local host and the local domain. Depending on the cache, it may be necessary to specifically include the local host.

The following considerations apply:

1. Network settings

- WTP must be able to make a network connection to the cache (try pinging the cache from the WTP machine).
- The cache must allow caching of data from the domain and the fully resolved host name for the WTP machine.

2. Cache storage size

- If supporting only one device, the size recommended by the cache manufacturer should be sufficient.
- If supporting multiple devices/networks which are accessing the same pages through WTP, there will be a variant of the resource for every device/network combination. The recommended cache size should be increased accordingly.

6.4.2 When is caching beneficial?

There are several situations where caching could be useful in a specific scenario. For example:

- WTP is being used as a portal to a limited number of Web sites/pages.
- There are multiple users using WTP as a proxy with the same device.
- The proxy will be transcoding large documents or images.
- The browsers and devices do not have client-side caching.

In general, for caching to be useful, the pages that are retrieved do not need to be completely limited to a set of sites but should be concentrated on a set of sites. For example:

- When same sites/pages are rarely hit: caching may not be so useful.
- When same sites/pages are always hit, that is restricted to certain sites such as a small intranet: caching is very useful.

- When same sites/pages are often hit, that is access allowed to the entire Web but traffic concentrated on the company intranet: caching still very useful.

6.4.3 MEG groups and how they are used

The transcoding of a resource can also be split into several steps with intermediate caching points. This is accomplished by grouping the MEGs. The default transcoders were all placed in one group for this release so there is no intermediate caching by default. There is a toolkit sample that illustrates MEG groups and intermediate caching. See for example, Chapter 9, “Using the Toolkit” on page 157.

6.4.4 Sample configuration for Web Traffic Express

There is a redbook about the *IBM WebSphere Performance Pack: Caching and Filtering with IBM Web Traffic Express* (SG24- 5859) which describes in more detail the installation and configuration of WTE.

In this section we describe the necessary steps to configure IBM Web Traffic Express as the external cache server for this scenario.

There are two main settings that you have to configure on WTE. Enable cache and the directory/name file for caching. To change these settings, you need to open the Administration and Configuration utility and select **Cache Settings** from the list on the left frame.

Then, on the right frame, select the **Enable proxy caching** option.

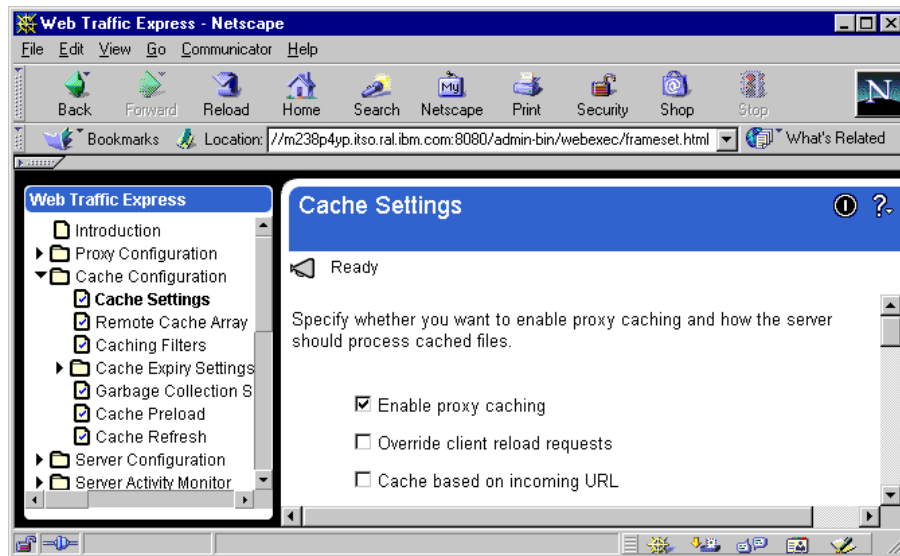


Figure 55. Enable proxy caching

Scroll down, on the same page, and enter the directory for the cached files and for the log file name. Then click **Submit** and restart the service on the Windows NT service panel.

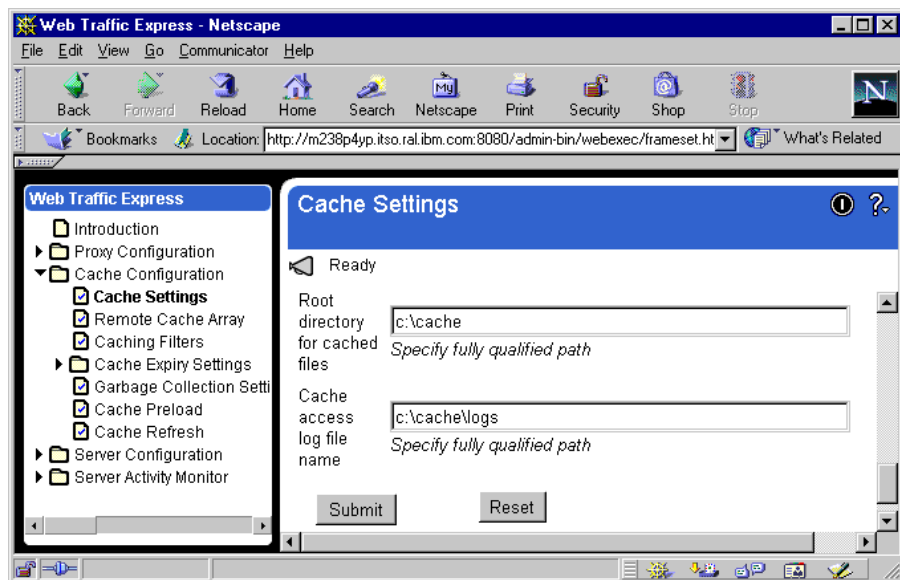


Figure 56. Caching directory and log file name

When the IBM Web Traffic Express service is restarted, you can try to access some pages on the Internet and check if files were created on the directory you selected above.

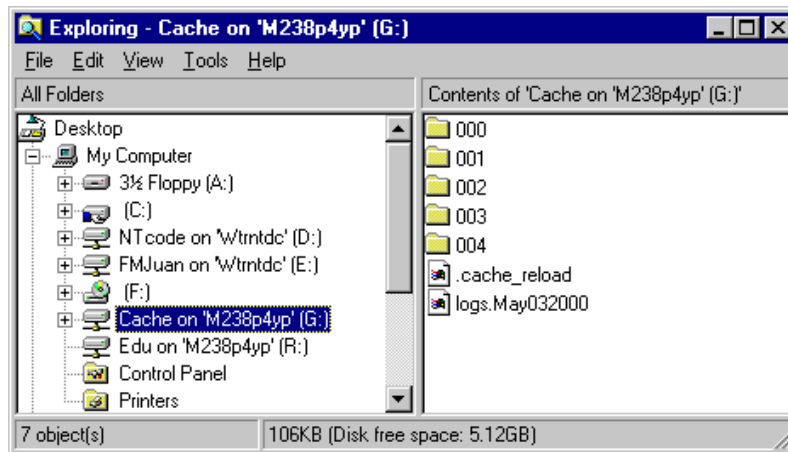


Figure 57. Cached files directory on WTE

Note

Do not delete these files. If you need to delete one of these files or directories, you will need to restart WTE before trying to cache anything.

6.4.5 Sample configuration for WinGate

We also installed and configured WinGate as the external cache server. It works fine, but read 6.4.1, “Considerations for configuring the external cache” on page 79, to make sure that you understand how to configure the external cache server correctly.

You can download an evaluation copy of WinGate from the Web site:
<http://www.wingate.com>.

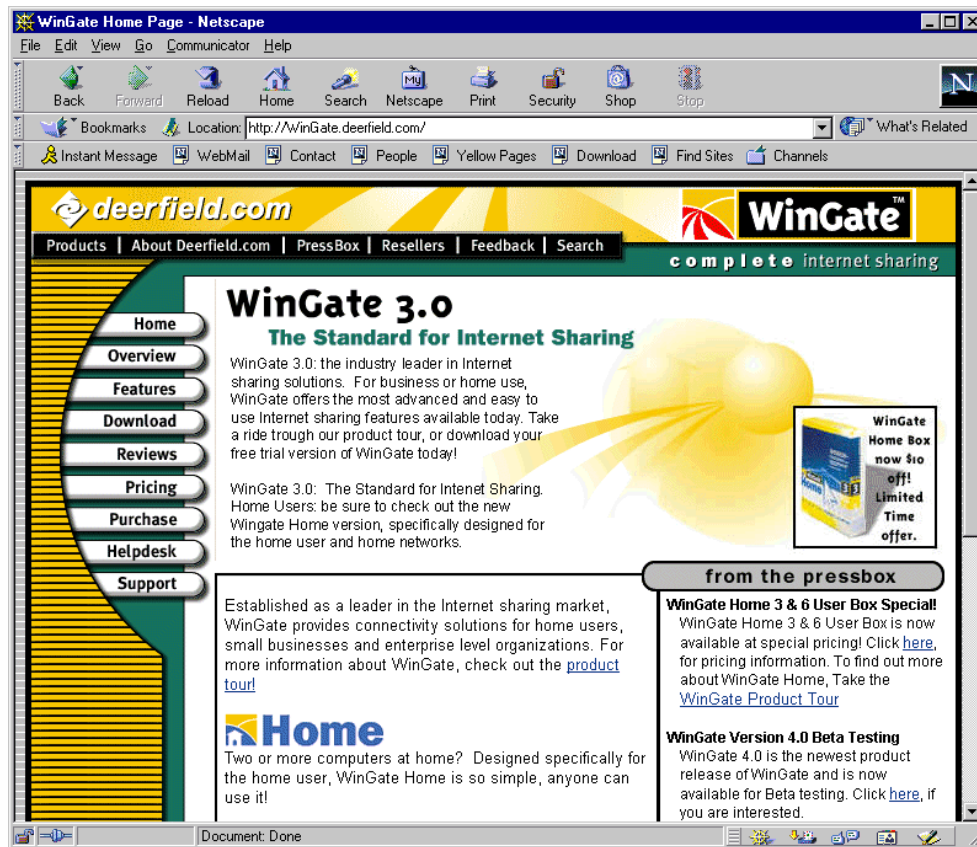


Figure 58. WinGate download on the Web

There are two main steps to install Wingate as the external cache server:

- Select the option to install as a WinGate server.
- Select the version option. In this scenario, we have installed the Pro (fully featured) version.

To configure it as the external cache, you only need to open the GateKeeper (administration utility) and make sure that these services are configured:

- DNS Service
- Caching
- WWW Proxy Server

We used the default for all options, but you have to add your adapter to the Bindings on the DNS Service and on the WWW Proxy Server. To open the

properties for that service, you need to click with the right mouse button on the name of the service and then select **Properties**.

On the second page (Bindings) you need to add your adapter to the list of bound adapters.

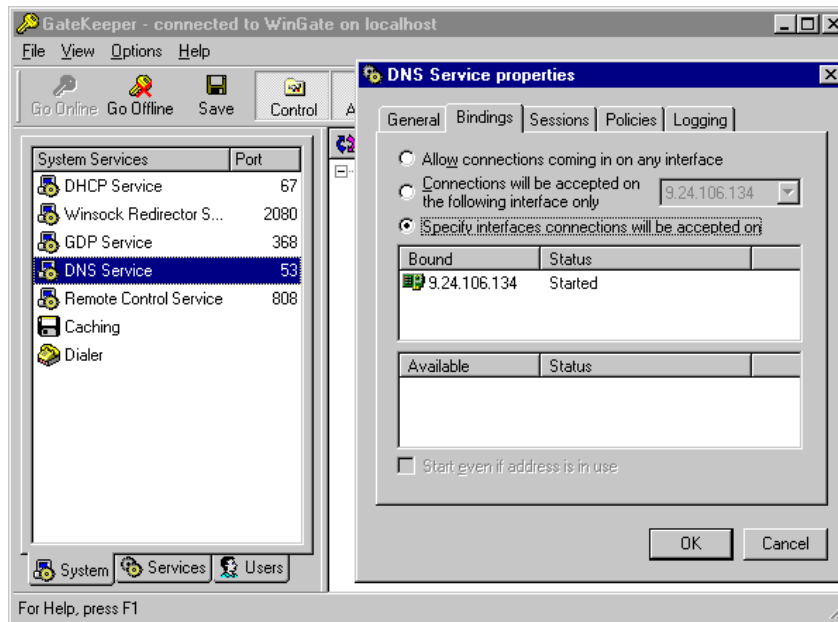


Figure 59. DNS Service properties - GateKeeper system page

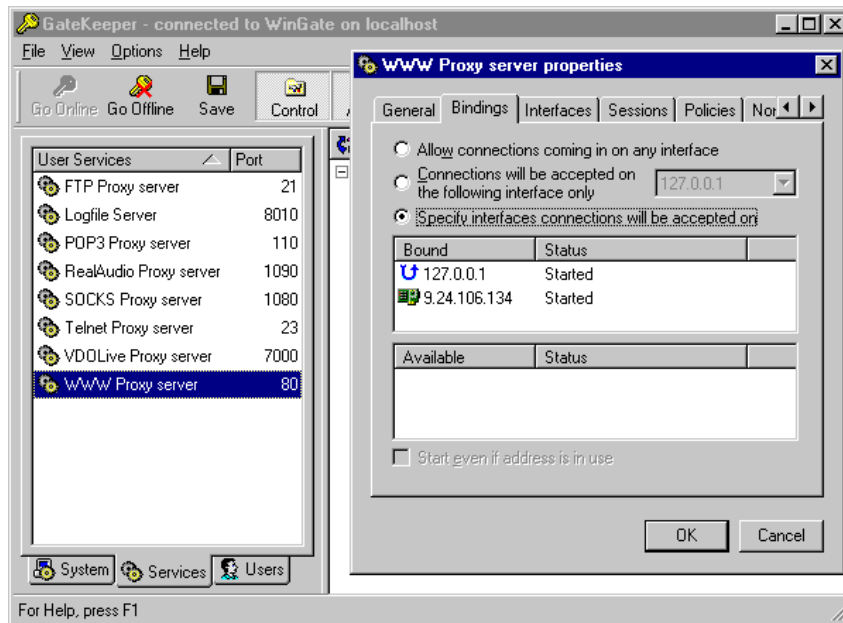


Figure 60. WWW Proxy server properties - GateKeeper system page

When you finish the configuration, click **Save** (diskette icon) to commit the changes to disk.

WinGate will be installed as an NT Service. It will be set to automatically start. You do not need to restart the service after finishing these configurations. Just close GateKeeper and begin surfing on the Internet to see if files will be added to the cache directory.

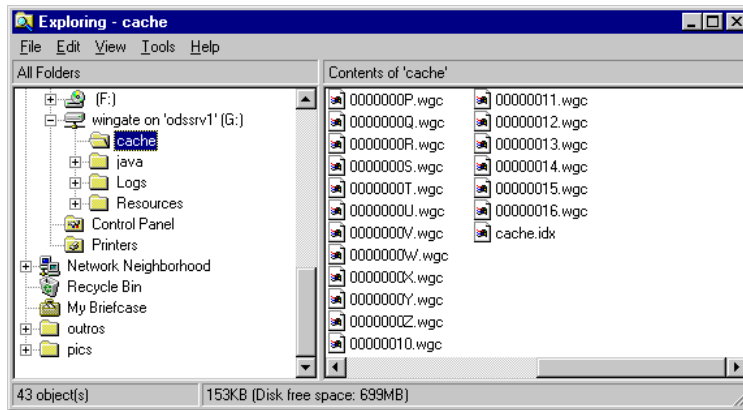


Figure 61. WinGate cached files directory

Note

Do not delete these files. If you need to delete one of these files, you will need to restart WinGate before trying to cache anything.

6.5 Tuning cache server

Most HTTP caching proxies use a calculation based on the Last-Modified date to determine the expiration of a cached document. Many caching proxies provide a means to control this. This means that an administrator can control how long the document stays in the external cache. If your users will view a lot of pages that do not change frequently, you might save processing time by increasing the length of time a document can reside in the cache before expiring.

6.6 Troubleshooting

Since most browsers do not provide access to the response headers, there is no client-side method of determining the source of the transcoded output (cached resource vs. new transcoded resource).

Checking the logs of the cache should provide information about hits and misses for transcoded resources.

If cache logs are not available or convenient, stop the IBM WebSphere Transcoding Publisher service and start the Request Viewer. It should list the cache plug-ins and sublayers (double-click on the folder to expand the tree).

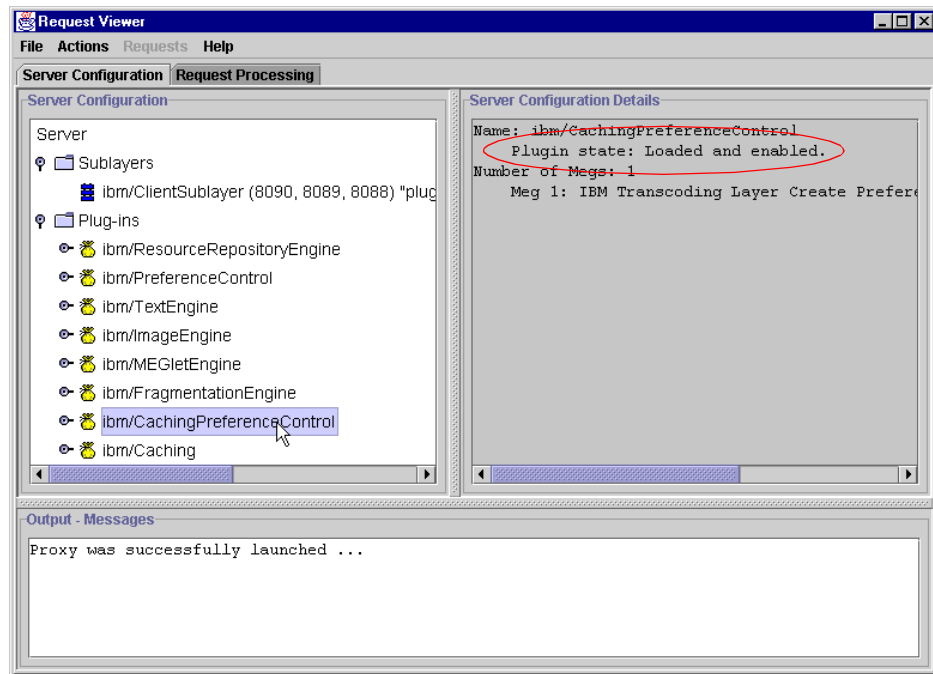


Figure 62. Request Viewer plug-ins - caching

Select the Request Processing tab to view the requests, go to the browser and try to access some sites/pages. The window should be similar to the one shown in Figure 63 on page 89.

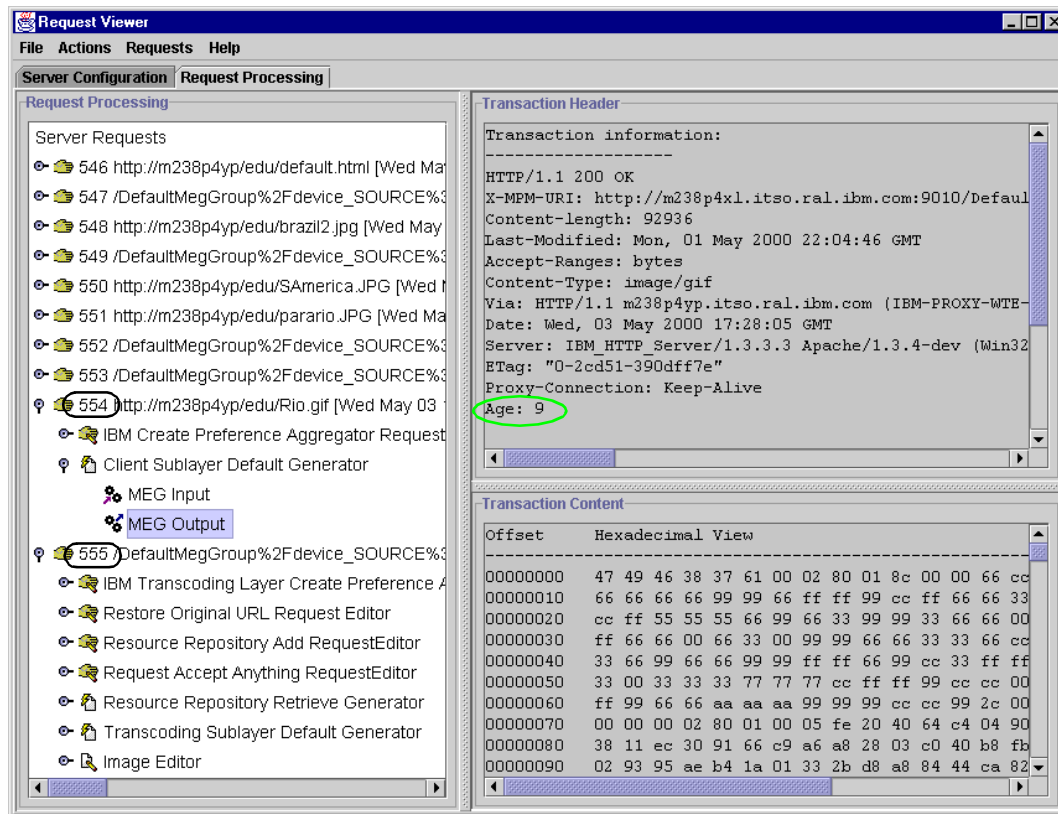


Figure 63. Requests from the browser and the external cache server

Figure 63 illustrates that for each request from the browser, beginning with “http://” as in number 554, there is a request from the cache server to WTP in order to get and transcode the page for that specific request (device/network preferences), beginning with /DefaultMegGroup/ as in number 555.

The next time this page is requested from the browser, WTP will get the page already transcoded from the external cache server and will send it to the browser. It is illustrated in the server request number 565 in Figure 64 on page 90.

Most caches, including WTE, will add an AGE: header which indicates the document was returned from a cache.

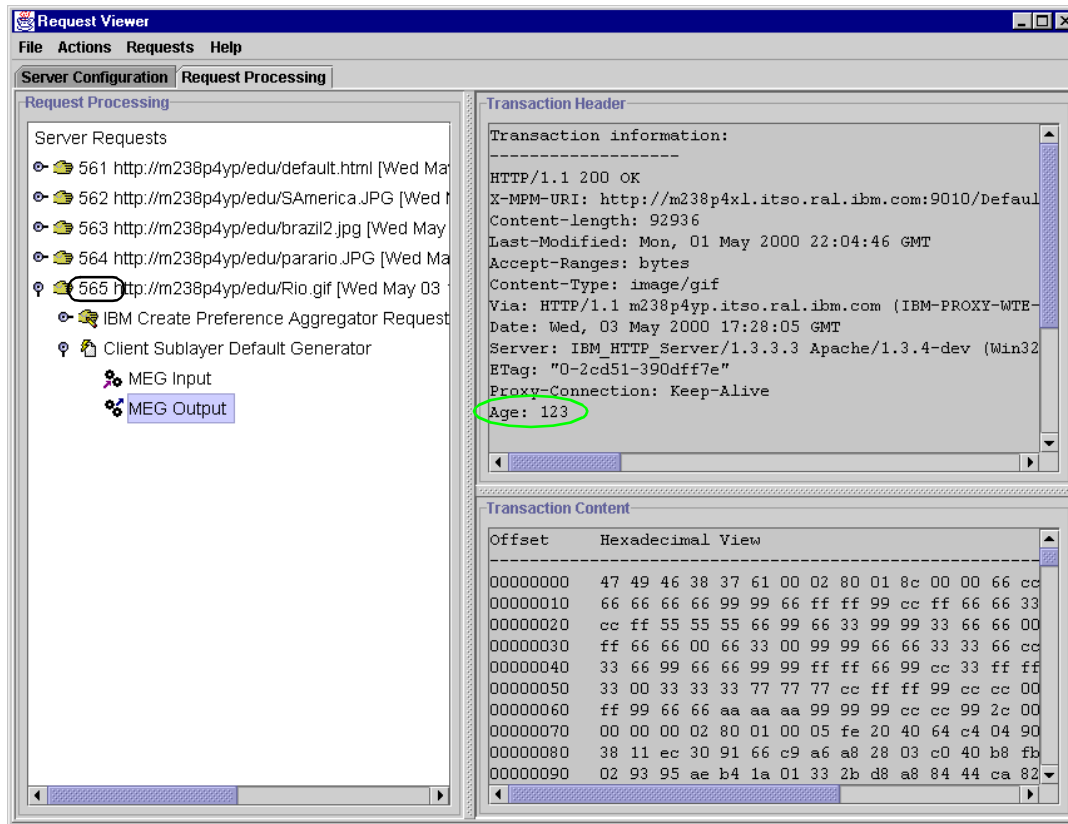


Figure 64. Requests from the browser only

In other words, a request from a client for a new resource will create two entries in the Request list, one for the client request, and a second for the transcoded output request. In this case, there is also an intermediate caching point, so two subsequent transcoding requests are visible.

A request from a client for a cached output will create only one entry in the Request list. No subsequent request appears for the transcoded resource.

There are several conditions on obtaining a cache hit:

1. Your subsequent request is not a page reload: If you are trying to verify functionality of a cache, you cannot use the Reload button in a browser, as this appends a directive to the request (in the form of an HTTP Request Header "Pragma: No-Cache"), which forces the request to be serviced by the original Web server.

To verify caching it is easier to close and open the browser again (all windows) and point to the same URL. Or you can use two machines with the same browser (both IE or Netscape) and send the request for a cacheable page from the first machine, then issue the same request from the second machine.

2. The page you are looking at must be cacheable. Certain Web sites label pages uncacheable with an explicit header of "Pragma: No-Cache" or "Cache-control: No-Cache". A page can also be uncacheable if it has expiration information (as in the case with many dynamic pages, very common with main pages for portals like Yahoo or the main page of large sites) then the cache considers the content as dynamic (produced at request time) and does not cache it.
3. Make sure that your caching proxy server can cache files for the local domain.

There are other problems with corrective actions listed below:

- Local hostname not fully resolved:
 - Try pinging the fully-qualified hostname of each one of the machines, including the browser and the external cache server. You may need to configure the Windows NT network settings and include the domain name or check the DNS configuration.
 - You can also add an entry to a specific machine to the Hosts file in the `\winnt\system32\drivers\etc` directory.
 - In the case of multiple hostname machines, ignore warning, but cache should be in the same domain.
- Cache must be within the same firewall as the WTP server.
- Some caches will never cache local files, so WTP and cache might need to be on two different machines.
- The URL in the cache log should contain the name of the MEG groups for all of the enabled MEGs, preference values that are used to determine the variants and the original URL.

Example:

```
http://M238P4XL.itso.ral.ibm.com:9010/DefaultMegGroup%2Fdevice_SOURCE%3DNT.Netscape45%2Fuser_SOURCE%3Duser_default%2Fnetwork_SOURCE%3Ddefault%2FlocalPort%3D8088%2FUA_pixels%3Dnull%2FUA_color%3Dnull%24%57%42%49%24http:www.pc.ibm.com/us/index.html
```

Chapter 7. Transcoding with WAS filters

This chapter describes how to use and configure IBM WebSphere Transcoding Publisher (WTP) as a MIME-type filter in the IBM WebSphere Application Server (WAS). We provide an overview and show you how to enable your servlets and JavaServer Pages applications to run this function. Two sample scenarios are included to illustrate the steps required to transcode MIME-types text/html and text/xml.

7.1 Transcode at the source

In this section we illustrate why and how IBM WebSphere Transcoding Publisher has been implemented to run as a filter in the IBM WebSphere Application Server.

Note

IBM WebSphere Transcoding Publisher Version 1.1 requires IBM WebSphere Application Server Version 2.03 when running as a WAS filter.

7.1.1 Why transcode at the source

When you choose transcoding at the source, IBM WebSphere Transcoding Publisher (WTP) runs at the same location as the data that will be transcoded. In this way, no proxy configuration is needed and the operation is totally transparent to the clients.

When Transcoding Publisher (WTP) is configured to run as a WAS filter, it allows for the implementation of available security options. For example, Secure Sockets Layer (SSL) and other encryption algorithms can be used over network links and it has limited exposure of data that is not transcoded. In this environment, WTP allows transcoding of data that will be encrypted when it flows over the network link, so the transcoding can take place after the data has been generated but before it is encrypted by the Web server.

The control of both the content and transcoding is local. This approach is particularly well-suited to environments where control is desired over both the data and the way in which it is transcoded.

Finally, WTP administration in this model is easier since startup and shutdown are handled by the WebSphere Application Server (WAS).

7.1.2 When not to use transcoding at source

There are a few cases when you will need to consider the option of not running WTP as a WAS filter. For example, you should not use content source transcoding when:

- Using MEGlets that are not supported in WAS 2.03. It is expected that this support will be available in a future release.
- Using network type to port mapping.
- When deck fragmentation is required. It is expected that this function will be available in the filter model in a future release.
- Transcoding content from multiple sources is not supported.
- Debugging since the Request Viewer tool does not work for the filter option.
- Need to run WAS 2.03 which is not supported on Linux and Windows 2000.

7.1.3 Transcoding with filters

Figure 65 on page 95 illustrates the WTP operation flow when running as filters in a WebSphere Application Server (WAS) environment.

The request and response flow can be summarized as follows:

1. The client sends a request to the Web server.
2. The Web server forwards the request to WAS and it resolves any alias (and clones) in the request.
3. The TranscodingPreamble servlet forwards the request information.
4. The user servlet runs and dynamically produces the markup text (html or xml). If there are other static resources, such as images, with some exceptions they are more likely to be returned by the underlying Web server.

Note: In the current WAS implementation, if you want to transcode everything that goes through a particular server, you would either need to arrange to have it served up by a servlet or use the proxy model.

5. WAS adds the two results (from the TranscodingPreamble and from the user servlet) and forwards the aggregate to the TranscodingFilter.
6. The TranscodingFilter servlet transcodes to the target output using the request information and based on the configuration values in the proper device profile.

Note

The network profile is not used when WTP is configured to run as WAS filters.

7. The transcoded response is returned to the client.

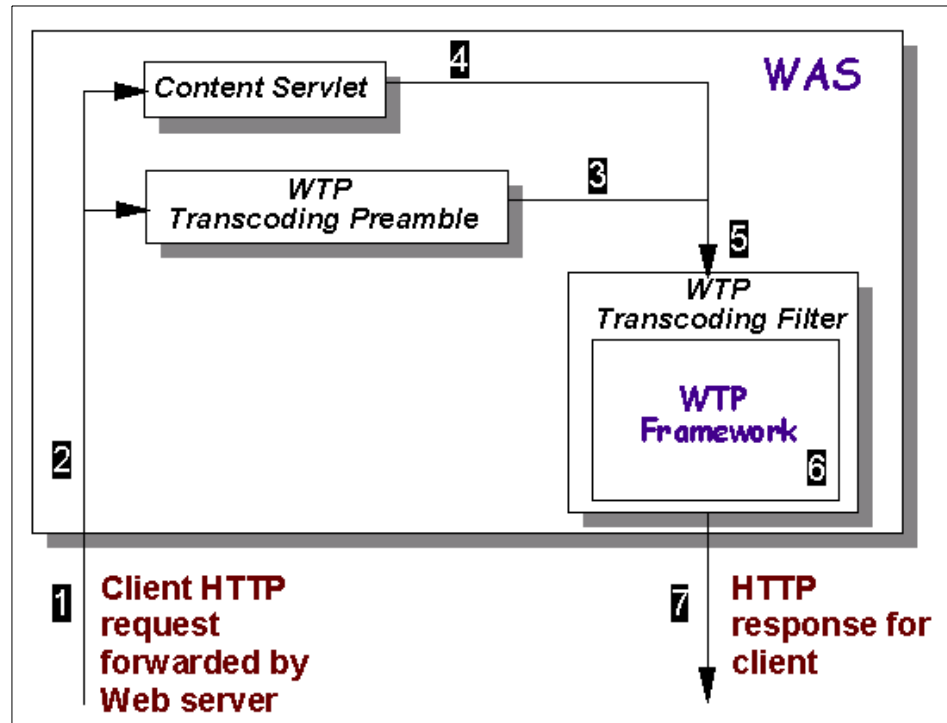


Figure 65. Transcoding with filters in WebSphere application server

7.1.4 Transcoding JavaServer Pages (JSP) content

JavaServer Pages (JSP) content can also be transcoded when running IBM WebSphere Transcoding Publisher as a WAS filter. However, if you enable transcoding for JSPs in WebSphere Application Server Version 2.03, the output from all JSPs will be transcoded.

For details on how to configure WAS to enable transcoding of JavaServer Pages (JSP) using WTP as a WAS filter see Chapter 11, “Transcoding Host Publisher application content” on page 235.

7.2 Configuring Transcoding Publisher as a WebSphere MIME-filter

Before you attempt to transcode any of your servlet content applications using the provided transcoding filters, you need to make sure of the following:

- You have installed the proper WAS supported Web server for example, IBM HTTP Server.
- You have installed a supported version of the WebSphere Application Server. See 1.2, “Hardware and software prerequisites” on page 2 for details.
- You have installed IBM WebSphere Transcoding Publisher to run as a MIME-filter in the WebSphere Application Server.

You should keep in mind that during installation, IBM WebSphere Transcoding Publisher can only be configured to run as a filter as long as the Transcoding Publisher installation process detects a supported version of the WebSphere Application Server.

Note

IBM WebSphere Transcoding Publisher Version 1.1 cannot be installed as a filter if there is no WebSphere Application Server already installed in your system.

If you did not install IBM WebSphere Transcoding Publisher to run as a filter because you did not have a supported version of WebSphere Application Server or for any other reason, you can always use the Server Setup option (see Figure 66) to reconfigure it to run as a filter. However, you will first need to install WAS to allow Transcoding Publisher to apply the required definitions and configuration settings to WAS.

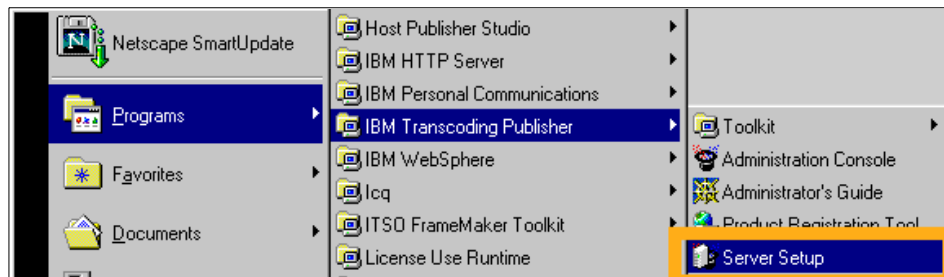


Figure 66. IBM WebSphere Transcoding Publisher Version 1.1 - Server Setup

For example, if you have previously installed Transcoding Publisher to run as a proxy server, you will use this option to set it up to run as a WAS filter (see Figure 67 on page 97).

Note

IBM WebSphere Transcoding Publisher cannot be configured to run as a proxy server and as a MIME-type filter concurrently.

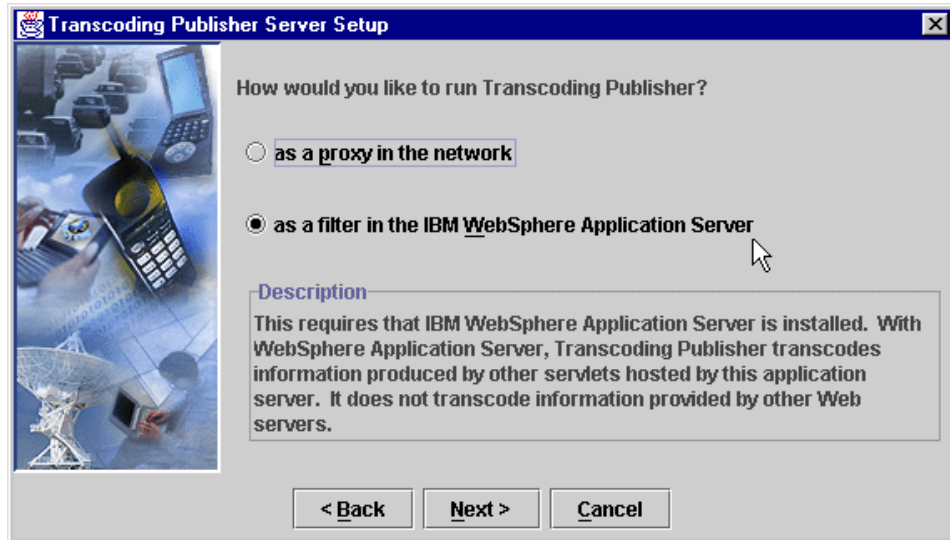


Figure 67. Configure Transcoding Publisher as a filter in WAS

The following process summarizes the required steps to configure IBM WebSphere Transcoding Publisher Version 1.1 to run as a MIME-filter in WebSphere Application Server:

1. Before starting the configuration, shut down WTP first and then WAS.
2. Start the Server Setup wizard as follows:
Select **Start->Programs->IBM Transcoding Publisher->Server Setup**.
3. Click **Next**.
4. Choose the option **as a filter in the IBM WebSphere Application Server** and click **Next** (see Figure 67 on page 97).
5. Click **Next** again and click **Finish** and the configuration is set.
6. Restart the WAS to pick up the changes made by this process.

7.2.1 Taking a look inside the WTP filter configuration

When you configure IBM WebSphere Transcoding Publisher (WTP) to run as a filter in WAS, the server setup process will do the following:

1. It selects *servlet* as the configuration to use.
2. It will locate WAS and verify its version. It must be a supported version, in this release 2.0.3 or 2.0.3.1.
3. It defines the WTP in the WAS configuration files (properties and other files). This means setting up the paths, setting filtering enabled and adding the servlets *TranscodingFilter* and *TranscodingPreamble* in the appropriate files. The updated WAS files are in the directory:

`<install path>/AppServer/properties/server/servlet/servletservice`

For example, see the updated `servlets.properties` file in Figure 68 which includes the Transcoding Publisher servlets added during WTP installation as a WAS filter. The transcoding servlets are:

- *TranscodingPreamble*. It obtains and forwards information about the client. It is chained together with the content servlet or JSP.
- *TranscodingFilter*. It is invoked as a WebSphere Application Server filter to transcode the servlet or JSP content using the client information forwarded by the *TranscodingPreamble* servlet and the user servlet or JSP content.

```
# Servlets added by the user
servlet.TranscodingFilter.code=com.ibm.transform.TranscodingFilter
servlet.TranscodingFilter.initArgs=InstallPath=C:\Program Files\IBMTrans
servlet.TranscodingFilter.description=
servlet.TranscodingPreamble.code=com.ibm.transform.TranscodingPreamble
```

Figure 68. Transcoding servlets in WAS `servlets.properties` file

7.3 WAS servlet configuration for transcoding

In this section we show how you would configure WAS and WTP to enable transcoding for your applications.

7.3.0.1 Prerequisites

Before you configure WAS to support filter transcoding, you need to verify the following:

1. You have successfully installed a Web server (for example, IBM HTTP server) and a supported version of WAS as a plug-in.

2. You have successfully installed and configured WTP running as a WAS filter as illustrated in 7.2, “Configuring Transcoding Publisher as a WebSphere MIME-filter” on page 96.
3. You have successfully deployed your WebSphere applications. User servlets and JavaServer Pages (JSP) are the supported addressable units but they can invoke other programs such as JavaBeans.

Once these steps have been accomplished, you are ready to configure filter transcoding support for your applications.

7.3.0.2 Configuration

The following list summarizes the required steps to enable filter transcoding in WAS and WTP to support your clients:

1. Use the Filtering option in WAS administration (see Figure 69) to associate a MIME-type (for example text/xml or text/html) to the provided WTP TranscodingFilter servlet. This definition allows the WTP filter servlet (TranscodingFilter) to get control each time a response with the specified MIME-type is generated. The WAS Administration Console can be started from the desktop or using port 9527 in the server URL.

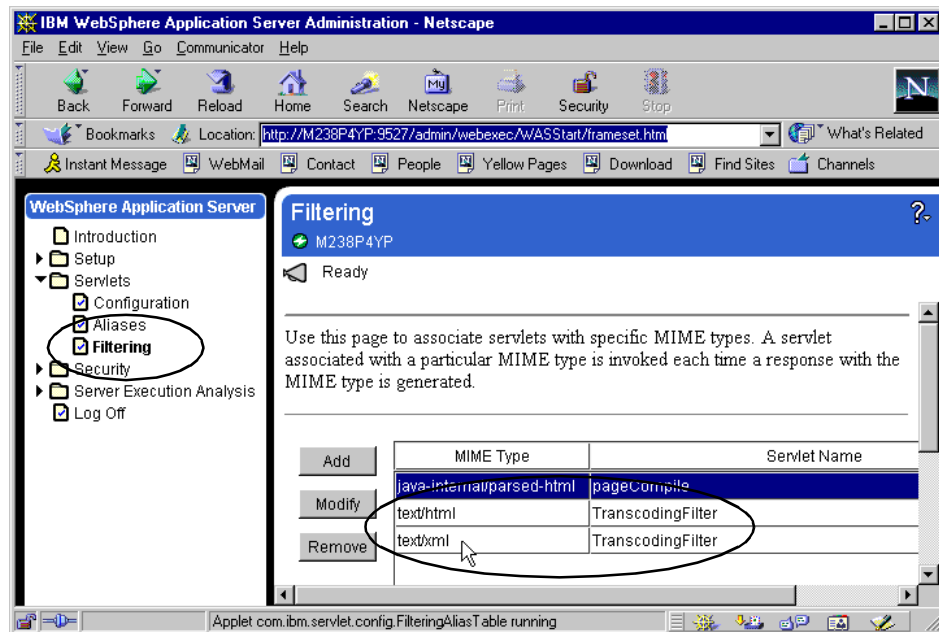


Figure 69. Enabling transcoding for MIME-types text/html and text/xml

2. Identify the servlets whose output will be transformed. For each of these servlets, use the Aliases option in WAS administration to define an alias adding TranscodingPreamble before the servlet name (see Figure 70 on page 100).

The alias enables the WTP servlet TranscodingPreamble to be chained to the user servlet. TranscodingPreamble captures information about the device type that is passed into the servlet, but which may not be passed out of the servlet. Notice that one entry per selected servlet is required.

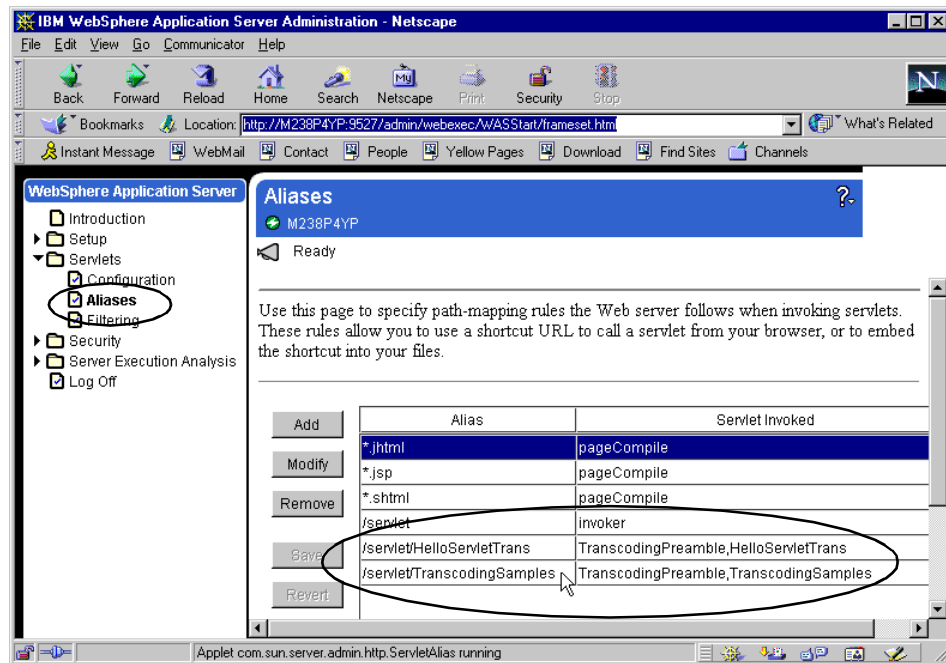


Figure 70. Chaining your servlets with TranscodingPreamble

3. If using JSPs in your application, use the Aliases option in WAS administration to define an alias by adding TranscodingPreamble before the pageCompile servlet name (see Figure 71 on page 101).

The alias enables the WTP servlet TranscodingPreamble to be chained with the generated JavaServer Page (JSP) servlets. Notice that all JSPs in the WAS server will use this option and therefore only one entry is required in this configuration.

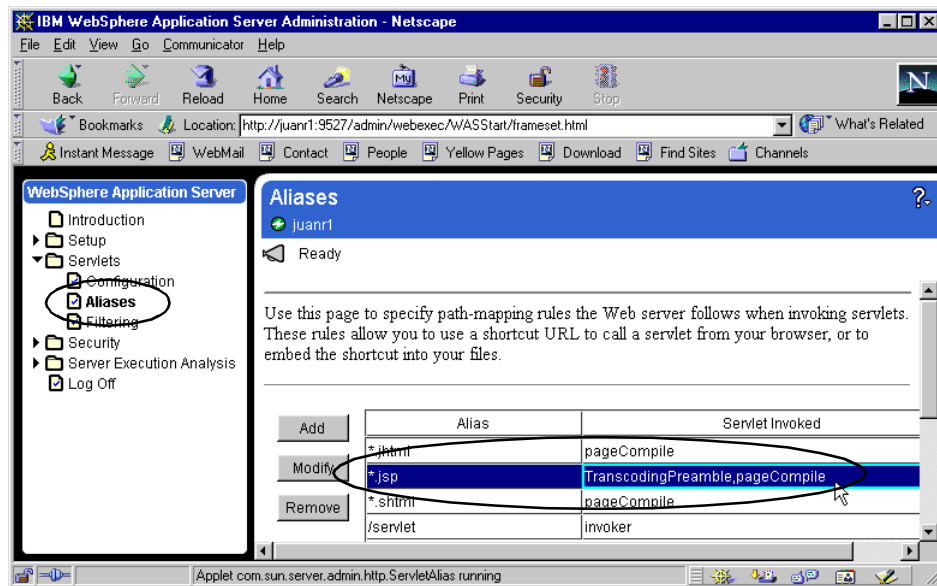


Figure 71. Chaining your generated JSP servlets with TranscodingPreamble

4. If needed, use the WTP Administration Console to configure or change any required device profiles for your clients (see Figure 72). Transcoding Publisher provides some default profiles as illustrated in Figure 72. If required, the Administration Console can be used to create new device profiles.

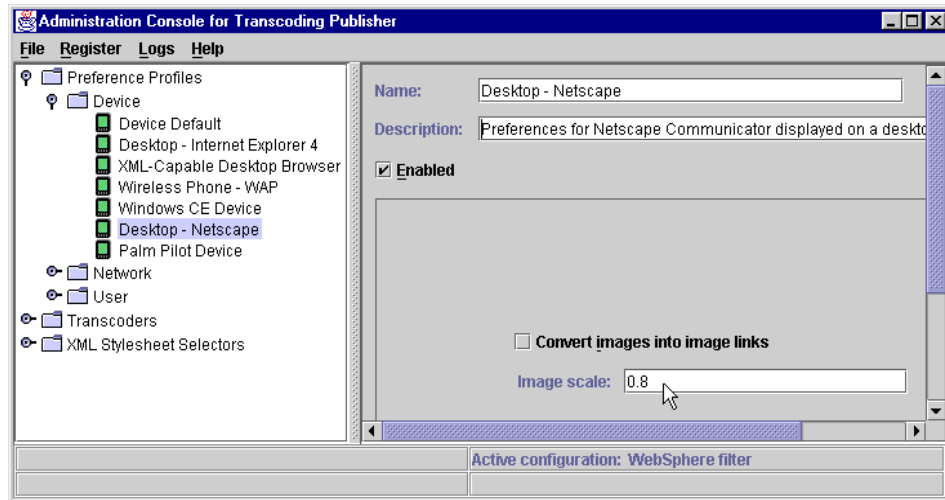


Figure 72. WTP Administration Console - Preference Profiles

5. If needed, use the WTP Administration Console to register and enable any required transcoders or XML Stylesheet Selectors. See 7.3.1, “XML stylesheet selection” on page 103 for details on how a registered XML stylesheet is selected by WTP.

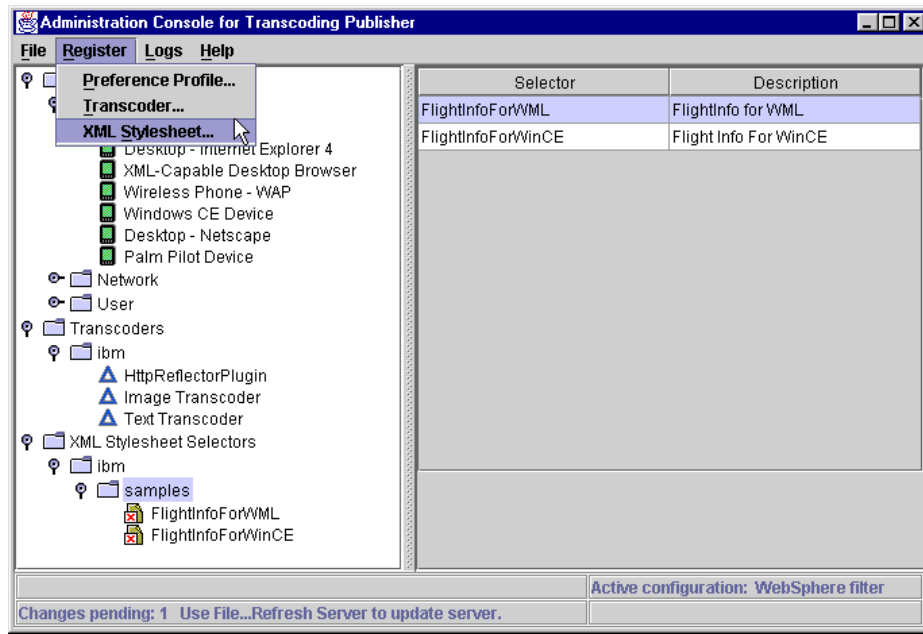


Figure 73. Register and enable transcoders and XML stylesheet selectors

6. After you finish the configuration, you may need to restart the WebSphere Application Server to make your changes take effect. For example, when you have just configured WTP to run as a servlet (WAS filter).

Use the **File->Refresh Server** option in the WTP Administration Console to update WTP when running as a servlet.

7.3.1 XML stylesheet selection

IBM WebSphere Transcoding Publisher (WTP) decides when to apply a stylesheet based on the DOCTYPE of the XML content and on the desired output content type. In addition, the optional XML stylesheet selection criteria is only necessary if the DOCTYPE and the desired output type are inconclusive due to a conflict.

Whether WTP is running as a servlet or as a proxy, the following rules are used by WTP to select the proper XML stylesheet for a specific client device:

1. The XML stylesheet must be registered. If required, use the WTP Administration Console to register a new XML stylesheet.

2. The XML stylesheet must be enabled. If required, use the WTP Administration Console to enable the XML stylesheet (see Figure 74 on page 104).
3. The XML stylesheet specifying the proper output content type is selected. For example, for a desktop browser and Windows CE, the stylesheet indicating output content type “text/html” will be selected. By the same token, for a WAP device, WTP will select the XML stylesheet indicating output content type “text/vnd.wap.wml” as shown in Figure 74 on page 104.
4. If more than one stylesheet specifies the same output content type, further information is required to determine which XML stylesheet to use for the proper client device. This extra required information is called the selection criteria.

7.3.1.1 Stylesheet selection criteria

Every stylesheet that is registered with Transcoding Publisher must have a unique set of selection criteria. Otherwise, the wrong stylesheet may be selected giving you unpredictable results.

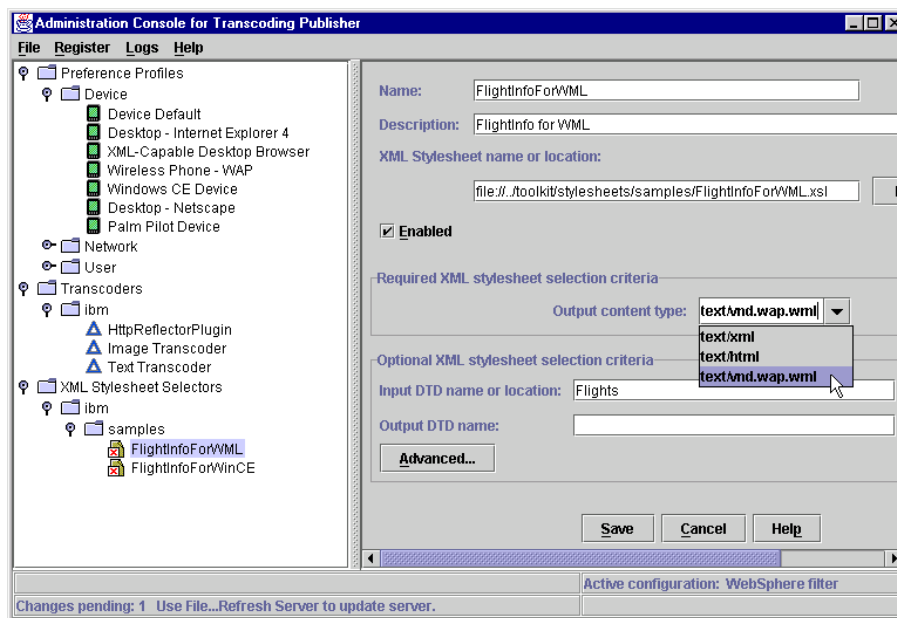


Figure 74. Enabling XML stylesheet selectors

The following selection criteria can be configured:

Required selection criteria

Each stylesheet selector must include selection criteria, which represent the output content type. Choose the correct output content type from the drop-down list or type a value in the entry box (see Figure 74). Valid values in this sample window are text/html, text/xml and text/vnd.wap.wml. However, the output content-type does not have to be one of the values listed in the combo-box since it is an editable combo-box and administrators can add their own content types. For example, you can add a stylesheet which does XML->HDML transformation. If you do this, you will need to specify “text/hdml” in the combo-box.

Optional selection criteria

This criteria is optional. Specify the file location or URL for the input DTD (document type definition). You can use the browse button or type the file location or URL, or a string that matches the name following the DOCTYPE keyword in the DTD. This stylesheet will be used only for XML documents that specify this same input DTD.

If the output type is text/xml, you can enter the name of the output DTD, or a string that matches the name following the DOCTYPE keyword in the DTD. The stylesheet will be used only for XML documents that specify this same output DTD.

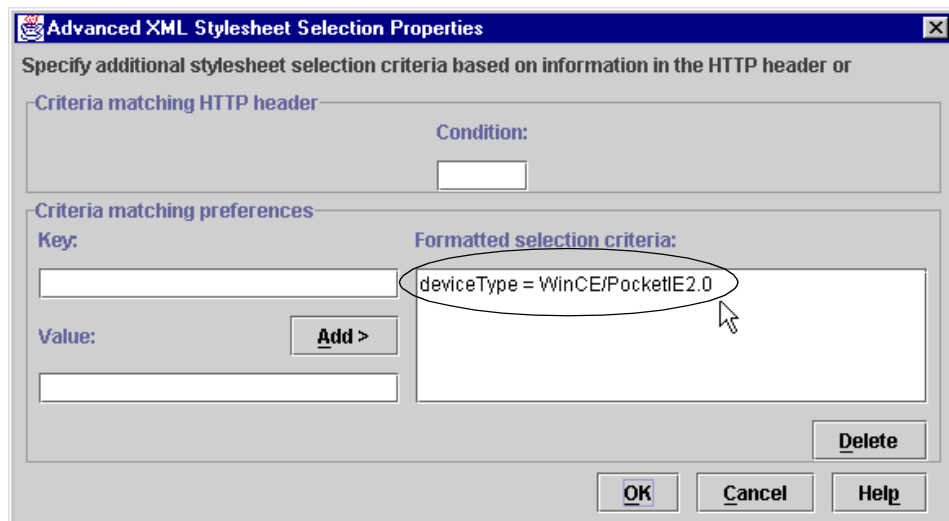


Figure 75. Advanced XML stylesheet selection criteria

Criteria matching HTTP header

Use the Advanced button (see Figure 74 on page 104) to specify additional criteria for the XML stylesheet. In the Criteria matching HTTP header box (see Figure 75), you can enter a formula that must be true for this stylesheet to be used. You can use any information in the HTTP header. For example, if you enter:

```
url=*chris/OrderStatusPhone*
```

then any document whose URL contains zero or more characters followed by "chris/OrderStatusPhone", followed by zero or more other characters would match this condition.

Criteria matching preferences

This option allows you to associate a stylesheet with a preference profile. Use this option to specify the criteria that will cause this XML stylesheet to be associated with a preference profile.

Transcoding Publisher selects a preference profile to use with each incoming document. You can add key/value pairs to your profiles, on the Advanced Preference Profile Properties panel, to help select a stylesheet. If these values have been specified for the profile, Transcoding Publisher will look for a stylesheet with matching information. To be selected, a stylesheet must match all of the key/value pairs that are entered as criteria matching stylesheets for the preference profile.

Note

A configured key/value pair must match the value in the particular device profile for a specific client device.

For example, if you added [device=Palm] to the preference profile (see Figure 76) for Palm Pilot devices, then you could add the same values to a stylesheet to cause it to be selected when an XML document is processed for a Palm Pilot device.

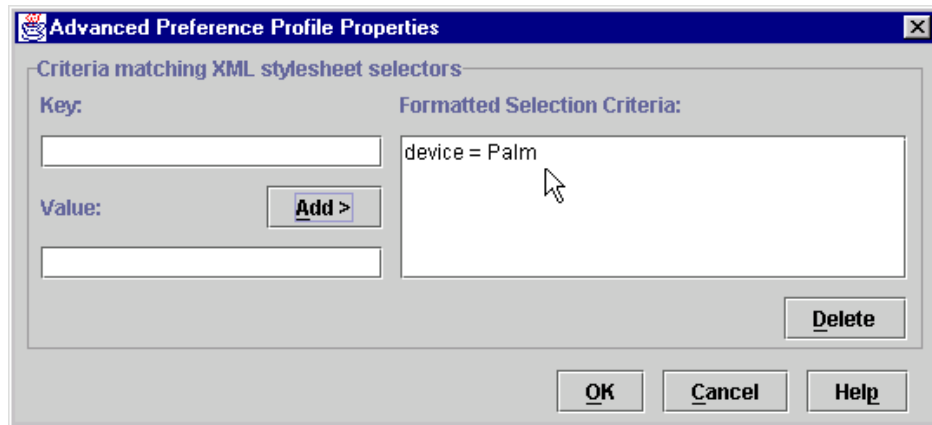


Figure 76. Device profile - stylesheet criteria matching selection

If a matching, eligible stylesheet is found, Transcoding Publisher will apply it to the XML document. An eligible stylesheet is one that has not been disqualified by failing to match other selection criteria, such as the output content type.

In the Criteria matching preferences box, you can enter one or more pairs specifying the name of a key and a corresponding value. These keys and values are not related to the incoming document; they can be any text strings meaningful to you.

Enter the values here and also on the Advanced Preference Profile Properties dialog box for the profile with which you want this stylesheet to be used. For a sample scenario using XML stylesheets, see 7.5, “Sample Scenario: transcoding XML content” on page 113.

7.4 Sample scenario: transcoding HTML content

In this section we include a sample configuration where IBM WebSphere Transcoding Publisher is used with a servlet application producing HTML and GIF images. We transcode the produced text/html and images based on the configuration options in the device profile for the specific client. Figure 77 illustrates this scenario, notice that WAS 2.03 chaining is in parallel.

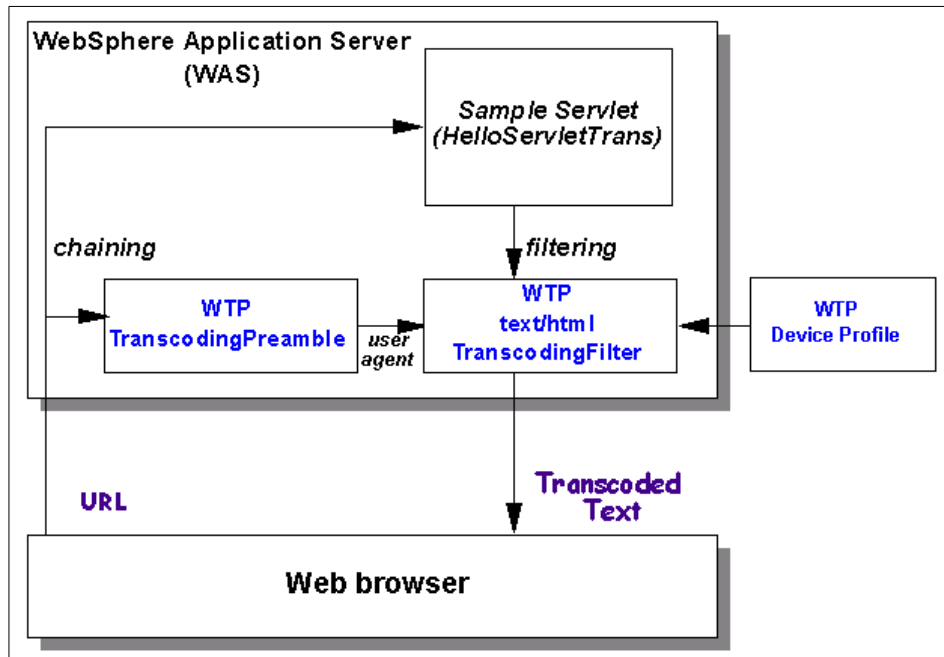


Figure 77. Sample scenario - text/html transcoding

The sample program we are using for this scenario is a servlet with name `HelloServletTrans` and it is listed in Figure 78 on page 109. When running WTP as WAS filters, you will need to chain the WTP `TranscodingPreamble` servlet with the user servlet. You also need to enable filtering in WAS for the MIME-type, `text/html` in this scenario. WTP uses the proper device profile (based on the detected user-agent) to execute the transcoding operation.

Note

When running WTP as WAS filters, the WTP network profiles are not used.

```

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloServletTrans extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {

        res.setContentType("text/html");
        PrintWriter out = res.getWriter();

        out.println("<HTML>");
        out.println("<HEAD>");
        out.println("<meta content='text/html; charset=iso-8859-1'>");
        out.println("<TITLE>Hello World Servlet</TITLE></HEAD>");
        out.println("<BODY LINK='\"#FFFFFF\"' ULINK='\"#FFFFFF\"' ALINK='\"#FFFFFF\"' bgcolor=WHITE>");
        out.println("<img src='\"/myapps/images/masthead.gif\"' WIDTH='\"640\"' height='\"32\"' ALIGN='\"top\"'>");
        out.println("<BIG><P>Hello World from Servlet</BIG>");
        out.println("</BODY></HTML>");
    }
}

```

Figure 78. Sample Servlet (HelloServletTrans)

The servlet produces html text and uses a file image (masthead.gif). When executing the servlet without transcoding, the page looks as shown in Figure 79.



Figure 79. The HelloServletTrans servlet without transcoding

7.4.1 Enable text/html transcoding

In this section we show how you would proceed to enable transcoding for this servlet when running IBM WebSphere Transcoding Publisher as a WAS filter. To accomplish this, we first review the procedure suggested in 7.3, “WAS servlet configuration for transcoding” on page 98 to make sure all prerequisites are in place for example:

1. In this sample scenario, the Web server and WAS have already been installed.
2. WTP has been installed and configured to run as a WAS filter.
3. The sample application (HelloServletTrans servlet) is properly running (without transcoding).

Next, we follow the suggested steps presented in 7.3.0.2, “Configuration” on page 99 to enable transcoding for this servlet for example:

1. In WebSphere administration, configure filtering for MIME-type text/html. The Transcoding Publisher servlet name for filtering is TranscodingFilter (see Figure 80 for details):
 - MIME-type: text/html
 - Servlet Name: TranscodingFilter

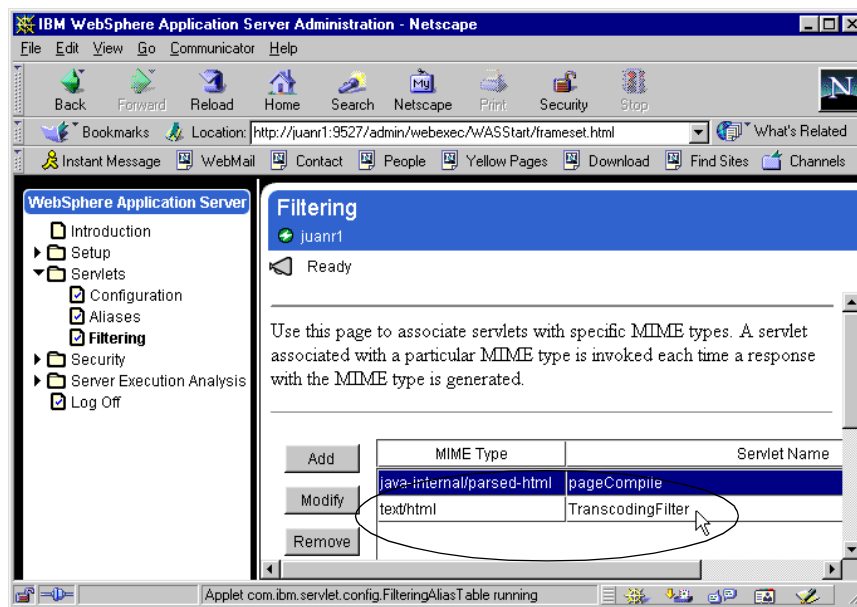


Figure 80. Enabling transcoding WTP filter for MIME-type text/html

2. In WebSphere administration, configure the alias for the chain with the Transcoding Publisher servlet (TranscodingPreamble) and the sample servlet HelloServletTrans. The following entry is added (see Figure 70 on page 100 for details):
 - Alias: /servlet/HelloServletTrans

- Servlet Invoked: TranscodingPreamble,HelloServletTrans

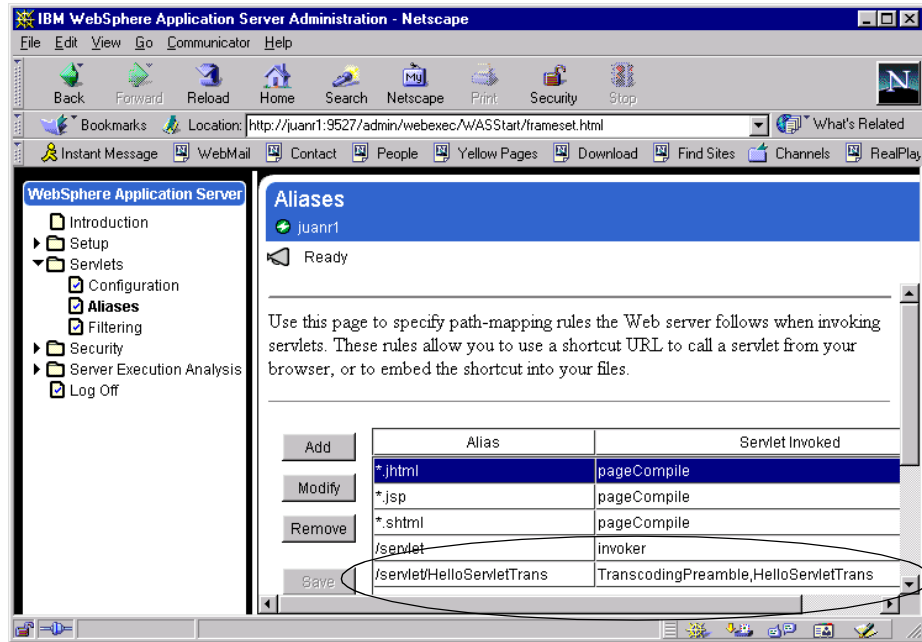


Figure 81. Chaining application servlet and WTP TranscodingPreamble servlet

- There are no JSPs in this scenario. For details on how you would configure transcoding for JSPs see Chapter 11, "Transcoding Host Publisher application content" on page 235.
- In this step you configure the profiles you require to support your clients. For example, the device profile for the Netscape browser is updated to use an image scale of 0.5 as illustrated in Figure 82 on page 112.

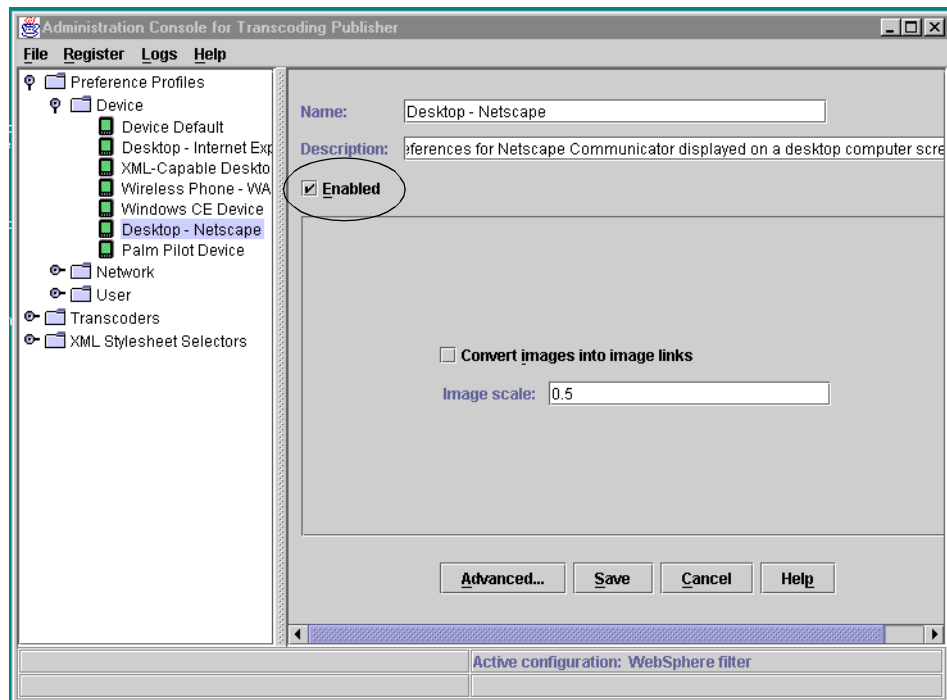


Figure 82. WTP Administration Console - device profile

5. The sample servlet does not produce XML in this scenario.
6. Use **File->Refresh Server** option in WTP Administration Console to pick up the new configuration values.

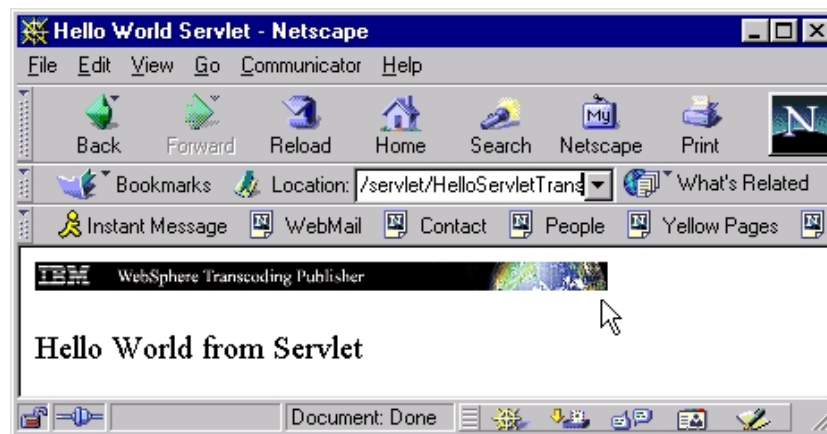


Figure 83. The HelloServletTrans servlet with filtering

Figure 83 on page 112 shows the results after invoking the WTP filter and using the associated device profile configuration. In this scenario, the image dimensions in the markup text have been scaled and this reduces the displayed image size.

7.5 Sample Scenario: transcoding XML content

In this section we include a configuration where IBM WebSphere Transcoding Publisher is used with a servlet application producing XML text and transcoding to HTML so the application can be accessed from other devices. This is, for example, the case where you have an XML application that needs to be transcoded because your Web browser does not support XML but only HTML. Figure 84 illustrates this scenario. Notice that WAS 2.03 chaining is in parallel.

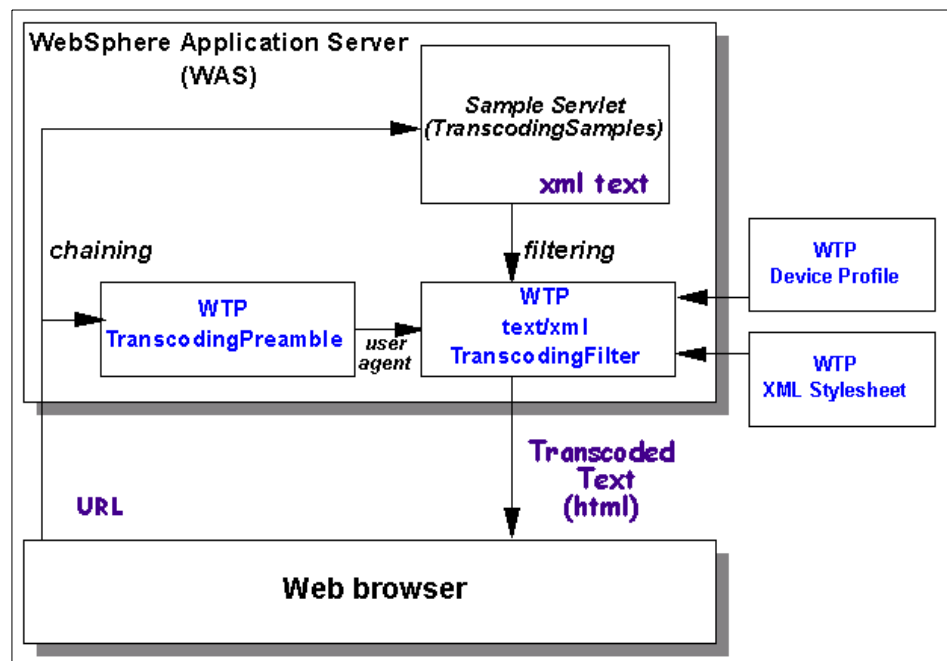


Figure 84. Sample scenario - text/xml transcoding

The sample program we are using for this scenario is provided by WTP in the `\toolkit\misc\` subdirectory. Its name is `TranscodingSamples` and it is listed in Appendix A, “TranscodingSamples sample program” on page 301.

7.5.1 Sample TranscodingSamples

The sample TranscodingSamples program can be run as servlet in a WAS environment or as a MEGlet in the WTP proxy model. It dynamically determines the MIME type based on the file extension passed in the URL request (see Figure 85 on page 114 for details). Therefore, it can produce XML pages, HTML pages, or even images based on the configuration in the RequestableSamples.prop properties file.

For example, when running as a servlet to produce XML content:

1. The URL to invoke the TranscodingSamples servlet is
`http://<server-name>/TranscodingSamples?name=FlightInfo.xml`
2. The servlet takes a parameter of the form name=FlightInfo.xml where FlightInfo.xml is an entry in a property file with name RequestableSamples.prop. For security reasons the sample program will only load xml files with a defined entry in this property file.
3. The RequestableSamples.prop file resides in the /toolkit/ subdirectory.

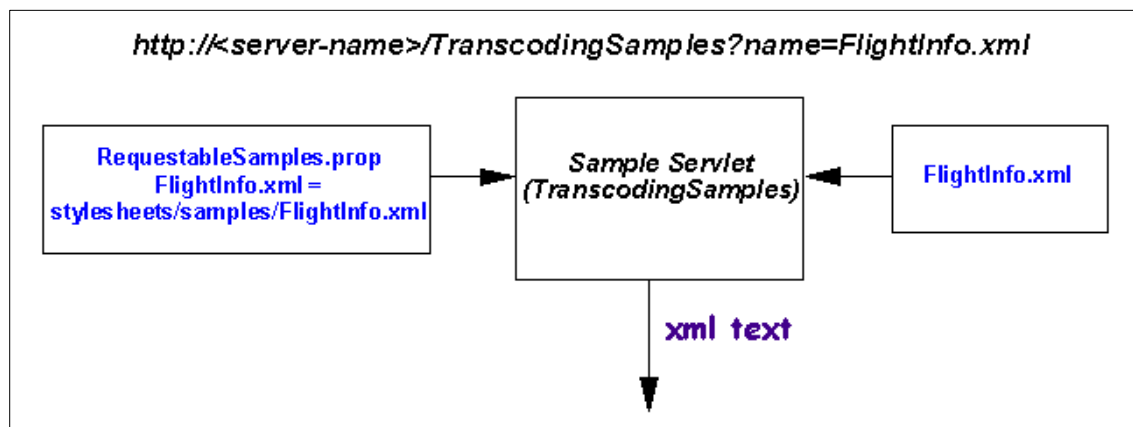


Figure 85. Sample servlet TranscodingSamples

4. The TranscodingSamples servlet requires an initial parameter InstallPath that must be defined in the WAS servlet configuration. For details on how you provide this parameter see Figure 88 on page 117.
5. The servlet gets a list of possible files it will accept from the RequestableSamples.prop file. In this scenario there is one xml file, FlightInfo.xml as shown in Figure 86 on page 115. FlightInfo.xml is located in the \IBMTrans\toolkit\stylesheets\samples\ subdirectory.


```

# RequestableSamples.prop
#
# List of requestable/transcodable sample files in the Transcoding Publisher toolkit.
# Files may be requested via http://host/servlet/TranscodingSamples?name=key in either the
# proxy or WebSphere servlet configuration once the TranscodingSamples servlet/MEGlet
# is installed.
#
# Format of this file:
#
#   key = location
#
# where location is relative to the location of this file. For security reasons, *only*
# files listed in this property file will be delivered by TranscodingSamples.

FlightInfo.xml = stylesheets/samples/FlightInfo.xml

```

Figure 86. Sample RequestableSamples.prop file

6. The content of FlightInfo.xml is shown in Figure 87 on page 115.

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Flights [
  <!ELEMENT Flights (AirCarrier, FlightNumber, FlightSegment)*>
  <!ELEMENT FlightSegment (DepartingInfo, ArrivingInfo)?>
  <!ELEMENT DepartingInfo (City,Date,Scheduled,Status?,Time?,Terminal?,Gate?,OffGateTime?,OffGroundTime?)?>
  <!ELEMENT ArrivingInfo (City,Date,Scheduled,Status?,Time?,Terminal?,Gate?,Baggage?,OnGroundTime?,OnGateTime?)?>
  <!ELEMENT FlightNumber (#PCDATA)>
  <!ELEMENT AirCarrier (#PCDATA)>
  <!ELEMENT Date (#PCDATA)>
  <!ELEMENT City (#PCDATA)>
  <!ELEMENT Scheduled (#PCDATA)>
  <!ELEMENT Time (#PCDATA)>
  <!ELEMENT Status (#PCDATA)>
  <!ELEMENT Terminal (#PCDATA)>
  <!ELEMENT Gate (#PCDATA)>
  <!ELEMENT Baggage (#PCDATA)>
  <!ELEMENT OffGroundTime (#PCDATA)>
  <!ELEMENT OffGateTime (#PCDATA)>
  <!ELEMENT OnGroundTime (#PCDATA)>
  <!ELEMENT OnGateTime (#PCDATA)>
]
<Flights>
  <AirCarrier>AA</AirCarrier>
  <FlightNumber>700</FlightNumber>
  <FlightSegment>
    <DepartingInfo>
      <City>DFW</City>
      <Date>31Mar</Date>
      <Scheduled>0630A</Scheduled>
      <Status>ON TIME</Status>
      <Time>0632A</Time>
      <Gate>C16</Gate>
      <OffGateTime>0644A</OffGateTime>
      <OffGroundTime>0632A</OffGroundTime>
    </DepartingInfo>
    <ArrivingInfo>
      <City>LGA</City>
      <Date>30Mar</Date>
      <Scheduled>1040A</Scheduled>
      <Status>ON TIME</Status>
      <Time>1043A</Time>
      <Gate>D5</Gate>
      <Baggage>B</Baggage>
      <OnGroundTime>1043A</OnGroundTime>
    </ArrivingInfo>
  </FlightSegment>
</Flights>

```

Figure 87. Sample FlightInfoXML file

Note: You must be careful if you are planning to update the RequestableSamples.prop file. When updating this file make sure you use an editor that does not line wraps existing text to avoid corrupting the file. This is a standard Java properties file and therefore, not using the proper editor may cause a load failure of this file.

7.5.2 Enabling Transcoding (XML to HTML)

The sample servlet we are using in this scenario produces XML text and we now want to transcode the XML contents into HTML text. In this section we show you the suggested steps to enable WTP transcoding running as WAS filters for the MIME-type text/xml.

7.5.2.1 Suggested procedure

We recommend the following steps to enable transcoding for applications producing XML text:

1. If the servlet is already compiled, make the servlet available to WAS. For example copy the servlet class file to the WAS \AppServer\servlets\ subdirectory.
2. Configure the servlet so it takes any required input parameters. In this scenario, the TranscodingSamples servlet requires the InstallPath parameter that you configure in the servlet properties. In this sample scenario, the WTP install path is (see Figure 88 for details):

```
c:\Program Files\IBMTrans
```

Optionally, you would configure the servlet load options and you can also load/unload the servlet at this time if required for testing purposes. More information is also found in the product online documentation Developer's Guide.

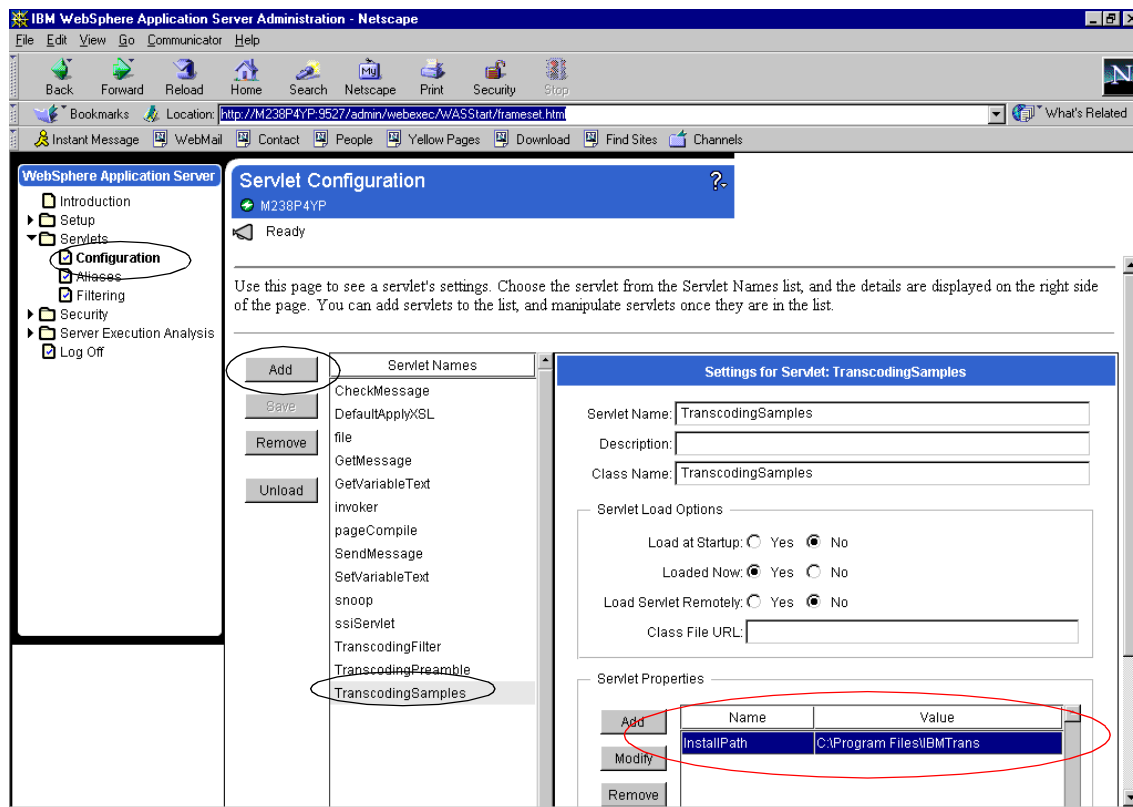


Figure 88. WAS administration - defining the TranscodingSamples properties

3. Configure chaining for the WTP TranscodingPreamble servlet and the application servlet. See Figure 70 on page 100 for details on how you configure this in WAS administration.
4. Enable WTP filtering for MIME-type text/xml. See Figure 69 on page 99 for details on how you configure this in WAS administration.
5. Restart WAS to accept the new configuration options.
6. In the WTP Administration Console, register the stylesheet for your client device. In this scenario, the client device is a desktop Web browser and we use the provided FlightInfoForNet_IE.xml stylesheet file. The XML stylesheet can be found in the \IBMTrans\toolkit\stylesheets\samples\ subdirectory and it is listed in Appendix B, "XML stylesheet (FlightInfoForNet_IE.xml)" on page 305.

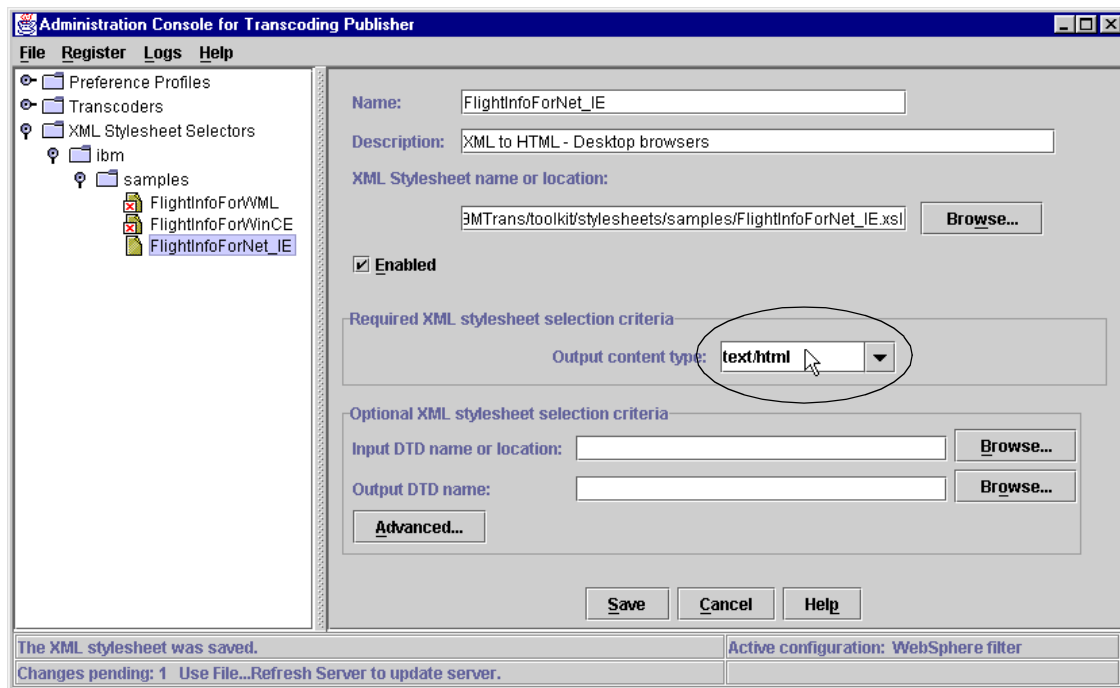


Figure 89. Registering the XML stylesheet (FlightInfoForNet_IE.xsl)

Notice that the output content type for the XML stylesheet is text/html. Since this is the only stylesheet for this output content type, there is no need to configure further selection criteria.

Note

If you have multiple XML stylesheets specifying the same output content type, you need to make sure that extra selection criteria are provided so the proper XML stylesheet is selected. Otherwise, the wrong XML stylesheet may be selected giving you unpredictable results.

7. Invoke the servlet using the following URL (replace <server-name> with your WAS server name):
<http://<server-name>/servlet/TranscodingSamples?name=FlightInfo.xml>

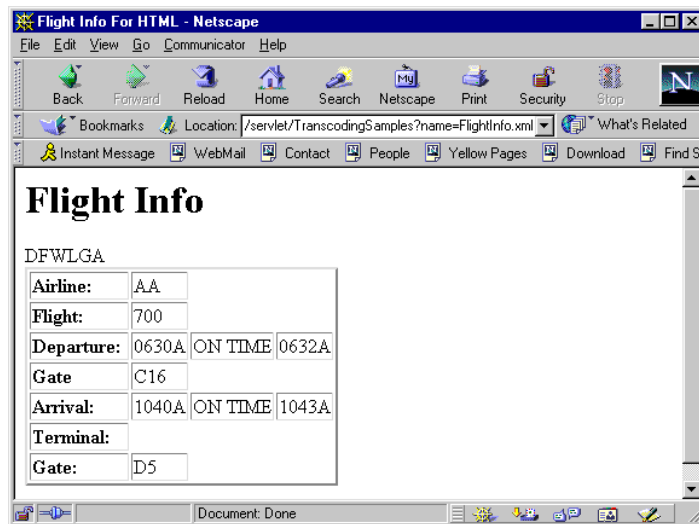


Figure 90. Receiving transcoded HTML text from the XML sample application

```
<html>
<title> Flight Info For HTML </title>
<body>
<p>

<h1>   Flight Info</h1>
</p>
<p>
<table border="2">
<tr>
<td><b>Airline: </b></td><td>AA</td>
</tr>
<tr>
<td><b>Flight: </b></td><td>700</td>
</tr>
  DFWLGA
<tr>
<td><b>Departure: </b></td><td>0630A</td><td>ON TIME</td><td>0632A</td>
</tr>
<tr>
<td><b>Gate </b></td><td>C16</td>
</tr>
<tr>
<td><b>Arrival: </b></td><td>1040A</td><td>ON TIME</td><td>1043A</td>
</tr>
<tr>
<td><b>Terminal: </b></td><td></td>
</tr>
<tr>
<td><b>Gate: </b></td><td>D5</td>
</tr>

</table>
</p>
</body>
</html>
```

Figure 91. Transcoded HTML text

When the servlet is called, it will generate XML text and translate it into HTML with the WTP TranscodingFilter servlet. Figure 90 on page 119 shows the transcoded results and Figure 91 on page 119 shows the HTML text.

Note: If you get the transcoded results in the wrong format, make sure you have properly configured the required selection criteria for the XML stylesheets so the proper stylesheet is used. For details about how you configure the XML stylesheet selection criteria see 7.3.1.1, “Stylesheet selection criteria” on page 104.

Chapter 8. Transcoding with JavaBeans

IBM WebSphere Transcoding Publisher (WTP) is also distributed as JavaBeans. In this model, Java programs such as servlets and JavaServer Pages (JSP) can invoke WTP transcoding capabilities directly from the user application. This chapter describes this capability of Transcoding Publisher in the JavaBean model. It also includes several scenarios that show you how to invoke WTP JavaBeans from your business application.

8.1 Overview

The transcoding capabilities of Transcoding Publisher have also been packaged as separate JavaBean components that can be used with other applications, such as Java servlets or JavaServer Pages.

The JavaBeans are a way of segregating the basic transcoding functions of Transcoding Publisher and making them available in a scaled down version that, while lacking some of the more complex interactions of the transcoding server, has the advantage of less processing overhead and easier implementation.

The JavaBean wrapper provides the transcoder with the same information about the system and request that it receives inside the Web Intermediaries (WBI) framework, so that it operates the same way in both contexts.

8.1.1 WTP JavaBean model

The JavaBean approach trades off flexibility for ease of use and less overhead. The idea behind the approach is to give the user a scaled-down version of the data transform functions to use because they did not want all the overhead of the WTP proxy model. An application service would be just as big as the proxy model in terms of code and configuration.

A transcoding bean is much smaller and simpler to understand. The trade-off is that the application has to know what the data is and therefore what transcoder to call. In addition, the caller may also have to provide additional information to the transcoder that would have been determined automatically in the full proxy environment. For example, the user-agent information that would have been provided by the HTTP request header may have to be supplied.

Some additional advantages are:

1. A JavaBean can be used as a building block in other applications (like JSP, etc.).
2. Improve performance by making local calls instead of making calls to some remote service.
3. A JavaBean fits in the JSP model without requirements of new directives.
4. Transcoding with JavaBeans does not have the overhead of a stand-alone service.
5. It also allows the user to have control over how long the function is instantiated.

8.1.2 WTP-provided JavaBeans

As illustrated in Figure 92, WTP JavaBean files are included at installation time. You will need to select the check box option in order to have the JavaBeans available during application development.

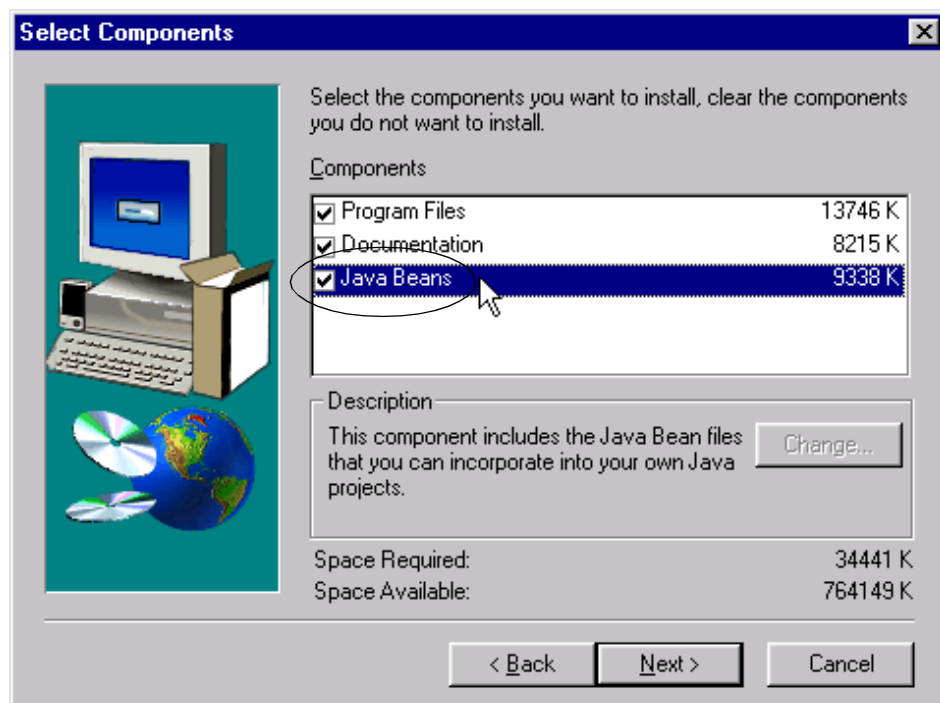


Figure 92. Including WTP JavaBean files in your system

For information about IBM WebSphere Transcoding Publisher installation see Chapter 3, "Installation" on page 41.

Transcoding Publisher (WTP) includes four JavaBeans transcoding options:

1. The `ImageTranscoderBean` converts images to different formats and performs other image transformations, such as scaling and color reduction. See 8.1.2.1, “Class `ImageTranscoderBean`” on page 123 for details.
2. The `HtmlReducerBean` simplifies HTML documents in various ways, including replacing images with textual links and stripping out objects or features that might not be supported by the target device. See 8.1.2.2, “Class `HtmlReducerBean`” on page 125 for details.
3. The `HtmlHandlerBean` converts HTML data into Wireless Markup Language (WML) in this release. See 8.1.2.3, “Class `HtmlHandlerBean`” on page 125 for details.
4. The `XmlHandlerBean` transforms XML data from one flavor of XML to another, based on the application of XML stylesheets. See 8.1.2.4, “Class `XmlHandlerBean`” on page 126 for details.

8.1.2.1 Class `ImageTranscoderBean`

This component wraps the WBI imageEditor MEG in a JavaBean callable service implementation.

There are three methods called service:

- `public java.io.InputStream service(java.util.Dictionary prefs, byte[] image)`

Parameters:

`prefs`: A map of the preference values to associate with the request. The keys for the preference values must be of the type `String` otherwise a `RequestRejectedException` will be thrown.

`image`: The image to transcode.

Return:

The streamed transcoded image result.

Throws:

`com.ibm.wbi.RequestRejectedException` - Thrown if an error occurs while servicing the input or the request is rejected by the underlying image engine.

`java.io.IOException` - Thrown if an error occurs while parsing the input/output stream.

- `public java.io.InputStream service(java.util.Dictionary prefs, byte[] image, int start, int length)`

Parameters:

`prefs`: A map of the preference values to associate with the request. The keys for the preference values must be of the type `String` otherwise a `RequestRejectedException` will be thrown.

`image`: The image to transcode.

`start`: The starting index in `image`.

`length` - Number of bytes in `image` to read.

Return:

The streamed transcoded image result.

Throws:

`com.ibm.wbi.RequestRejectedException` - Thrown if an error occurs while servicing the input or the request is rejected by the underlying image engine.

`java.io.IOException` - Thrown if an error occurs while parsing the input/output stream.

- `public java.io.InputStream service(java.util.Dictionary prefs, java.io.InputStream image)`

Parameters:

`prefs`: A map of the preference values to associate with the request. The keys for the preference values must be of the type `String` otherwise a `RequestRejectedException` will be thrown.

`image`: The streamed image to transcode.

Return:

The streamed transcoded image result.

Throws:

`com.ibm.wbi.RequestRejectedException` - Thrown if an error occurs while servicing the input or the request is rejected by the underlying image engine.

`java.io.IOException` - Thrown if an error occurs while parsing the input/output stream.

For more information see the product online documentation in [*<install-path>/toolkit/apidoc/index.html*](install-path>/toolkit/apidoc/index.html).

8.1.2.2 Class `HtmlReducerBean`

This component wraps the WBI ReducerHandler MEG in a JavaBean callable service implementation. The ReducerHandler MEG is responsible for performing HTML Simplification (or reduction) based on preferences and constraints in the device profile which describe the capabilities and limitations of the device or browser which displays HTML content.

- `public java.lang.String service(HttpPreferenceBundle prefs, String inputPage)` to transform an input page.

Parameters:

`prefs`: A map of the preference values to associate with the request. The keys for the preference values must be of the type `String` otherwise a `RequestRejectedException` will be thrown.

`inputPage`: The streamed input document.

Return:

The result document.

Throws:

`com.ibm.wbi.RequestRejectedException` - Thrown if an error occurs while servicing the input or the request is rejected by the underlying MEG editor.

`java.io.IOException` - Thrown if an error occurs while parsing the input/output stream.

For more information see the product online documentation in [<install-path>/toolkit/apidoc/index.html](install-path>/toolkit/apidoc/index.html).

8.1.2.3 Class `HtmlHandlerBean`

This component wraps the WBI HTMLHandler MEG in a JavaBean callable service implementation. The HTMLHandler MEG is responsible for transforming an HTML document into a Document Object Model (DOM), which can then be transformed by additional transcoding software into a different document. As of this writing, the HTMLHandler MEG supports transforming HTML into Wireless Markup Language (WML).

- `public java.lang.String service(HttpPreferenceBundle prefs, String inputPage)` to transform an input page.

Parameters:

`prefs`: A map of the preference values to associate with the request. The keys for the preference values must be of the type `String` otherwise a `RequestRejectedException` will be thrown.

inputPage: The streamed input document.

Return:

The result document.

Throws:

`com.ibm.wbi.RequestRejectedException` - Thrown if an error occurs while servicing the input or the request is rejected by the underlying MEG editor.

`java.io.IOException` - Thrown if an error occurs while parsing the input/output stream.

For more information see the product online documentation in [<install-path>/toolkit/apidoc/index.html](install-path>/toolkit/apidoc/index.html).

8.1.2.4 Class XmlHandlerBean

This component wraps the WBI XmlHandlerBean MEG in a JavaBean callable service implementation.

- `public java.lang.String serviceXml(java.util.Dictionary preferences, java.lang.String document)`

Parameters:

preferences: A map of the preference values to associate with the request. The keys for the preference values must be of the type `String` otherwise a `RequestRejectedException` will be thrown.

document: The input document.

Return:

The result document in "8859_1" encoding.

Throws:

`com.ibm.wbi.RequestRejectedException` - Thrown if an error occurs while servicing the input or the request is rejected by the underlying XmlHandler.

`java.io.IOException` - Thrown if an error occurs while parsing the input/output stream.

For more information see the product online documentation in [<install-path>/toolkit/apidoc/index.html](install-path>/toolkit/apidoc/index.html).

8.1.3 Invoking a WTP JavaBean

In order to use a WTP JavaBean, the calling application needs to create an instance of the WTP transcoder JavaBean; that is, it has to instantiate the JavaBean.

When using this model there is no WBI backbone, so there is no WBI overhead (or function). In this model, it is assumed that the caller, a user program in most cases, of the WTP JavaBean knows what transforms it needs to make and does not need WBIs logic to help make the determination. It also assumes that the calling application can instantiate the callable function just one time for the lifetime of the caller in order to minimize startup overhead.

Because the transcoding JavaBean has no external memory, the caller will have to pass in the information that is needed to do the proper transcoding. It is assumed that the caller has access to information relating to the current request and can pass it in as a “preference bundle”.

This preference bundle can include information from relevant HTTP headers. Through the preference bundle the transcoding JavaBean will have access to the relevant information that directs the type of transform that is done.

Note

When using the WTP JavaBean model, the user application must determine the client type (user-agent) and what transcoder JavaBean should be called.

8.2 Transform Tool sample program

The TransformTool.java sample program is a Java application and it does not require a browser or Web server to execute. It is available in source code as one of the tools supplied with Transcoding Publisher.

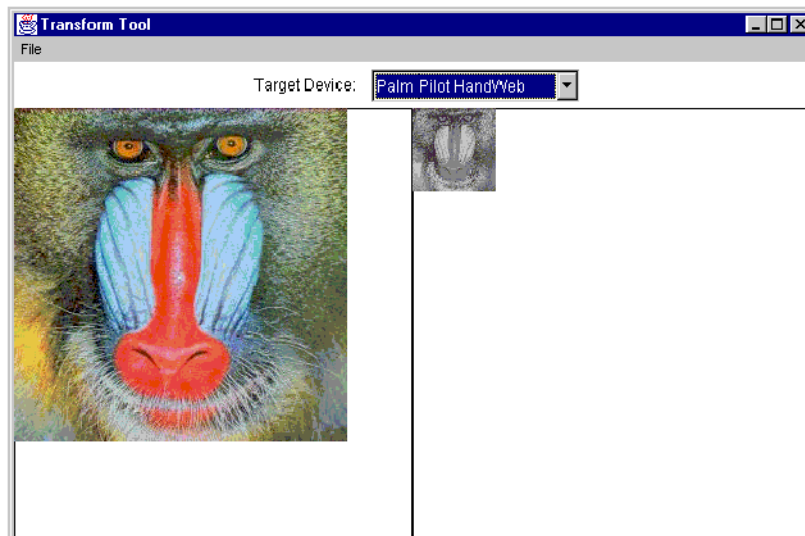


Figure 93. Using the Transform Tool

The Transform Tool is a quick way to preview how information is transcoded, with original and transcoded content displayed side by side. To perform its conversions, the Transform Tool uses the transcoding JavaBeans. The TransformTool.java program is located in the subdirectory:

`<install_path>/toolkit/beansamples/TransformTool.`

For more information about this tool, see Chapter 9, “Using the Toolkit” on page 157.

8.3 Image transcode sample program

This program is an example of how to use WTP JavaBeans with JavaServer Pages (JSP) technology. Since it is implemented using JSPs, it requires an application server such as the IBM WebSphere Application Server.

The ImageTranscoder sample demonstrates how to incorporate the ImageTranscoderBean in a JSP page that renders an image and then allows you to transcode it based on several different output devices. A sample JSP page is included for both JSP 0.91 (ImageTranscoder.jsp), which can be used with WebSphere Application Server Version 2.0.3 and JSP 1.0 (ImageTranscoder10.jsp), which can be used with servers such as Java Web Server 2.0. These files are located in:

`<install_path>/toolkit/beansamples/jsp`

along with an image file used in both JSP pages. To execute this sample program you may want to execute the following steps:

1. Copy the JSP file (or appropriate .jsp version supported by your server) from the WTP directory:

`<install_path>\toolkit\beansamples\jsp`

to the HTTP Server directory:

`<install_path>\htdocs`

2. Change the paths to the files:

- Instead of:

``

change to your own directory, for example:

``

- Instead of:

`file = new File("c:"+File.separator+"WWW"+File.separator+"HTML",
"ibmtrans.jpg");`

change to the same directory above, for example:

`file = new
File("c:\\WebSphere\\AppServer\\TxApps\\images\\ibmtrans.jpg");`

- Do the same changes with the output file.

3. Copy the jpeg file from the WTP directory:

`<install_path>\toolkit\beansamples\jsp`

to the HTTP Server directory:

`<install_path>\htdocs`

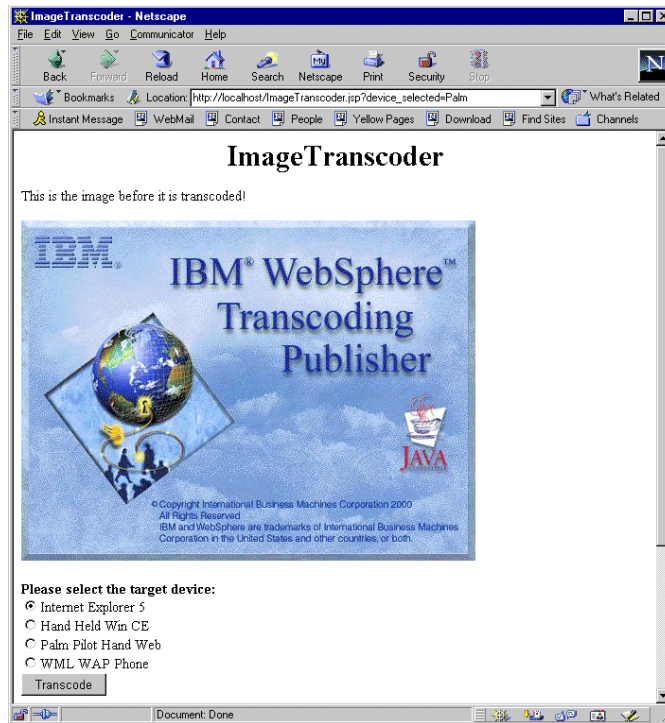


Figure 94. Using the ImageTranscoder sample program

4. To open this file from a browser enter for example:

`http://<server-name>/ImageTranscoder.jsp`

When viewed using a browser, as illustrated in Figure 94, the sample page shows the image that will be transcoded. After you select the target device, click the **Transcode** button to transform the image.

To obtain and see the transcoded version of the image, be sure to reload the page in the browser (see Figure 95 on page 131).

Note

When using the Netscape browser to reload a page, press the **Shift** key and select **Reload** from the menu bar.

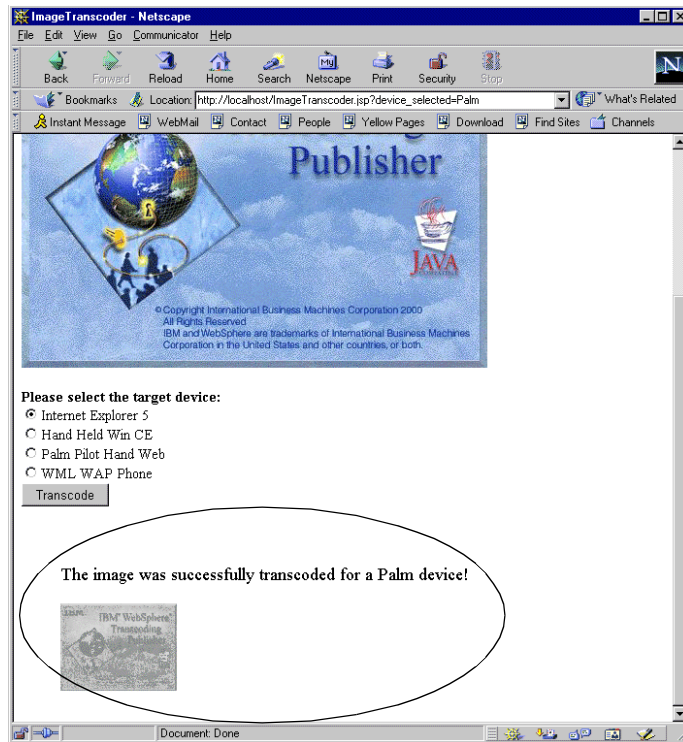


Figure 95. Transcoding an image for a Palm device

There are a few differences in syntax between JSP 0.91 and JSP 1.0 that you will need to consider, depending on which version you are using. Figure 96 illustrates these differences.

When declaring the usage of an instance of a JavaBeans component:

JSP 0.91: Use the Bean tag:

```
<bean name="itb" type="com.ibm.transform.bean.ImageTranscoderBean"...>
```

JSP 1.0: Use the `jsp:useBean` tag:

```
<jsp:usebean id="itb" class="com.ibm.transform.bean.ImageTranscoderBean"/>
```

When specifying JSP directives:

JSP 0.91:

```
<%@ language = "java"%>
```

```
<%@ import = "java.io.*", com.ibm.transform.bean.ImageTransformBean"%>
```

JSP 1.0: The page directive now collects several attributes.

```
<%@ page language = "java" %>
```

```
<%@ page import = "java.io.*", com.ibm.transform.bean.ImageTransformBean"%>
```

To declare class-wide variables for the servlet class:

JSP 0.91:

```
<script runat=server>
```

```
    // code for class-wide variables and methods
```

```
</script>
```

JSP 1.0:

```
<%|
```

```
    // code for class-wide variables and methods
```

```
%>
```

Figure 96. Differences between JSP 0.91 and JSP 1.0

8.4 Using the WTP JavaBeans

This section describes several considerations to keep in mind when using the Transcoding Publisher JavaBeans, for example:

- Defining the classpath and path information
- Specifying the input parameters required by the JavaBeans
- Special considerations for Linux users

8.4.1 Defining the classpath and path

Before you can use the transcoding JavaBeans or compile the JavaBeans-related samples, you need to ensure that the following information is added to your classpath (note that all of the items listed here are relative to the Transcoding Publisher installation directory):

- `.\beans\magicbeans.jar`
- `.\lib\xerces.jar`
- `.\lib\ras.jar`

- .\lib\xalan.jar
- .\lib\xpath.jar
- .\lib\js.jar
- .\lib\HTMLParse.zip
- .\lib\NetRexxC.zip
- .\lib\NetRexxR.zip

You will also need to update the path to include the Transcoding Publisher bin directory. In addition, you may also want to look at the provided TransformTool.bat file in the Transcoding Publisher installation directory.

8.4.2 Specifying input parameters

When running in the larger context of Transcoding Publisher, transcoders have access to mechanisms that can determine which transformations need to be performed and whether more than one transcoder should be involved in a given transaction. However, the transcoding JavaBeans do not have similar support. Due in part to their lightweight nature, the JavaBeans require that information about the transcoding environment be explicitly provided.

The following types of information must be specified:

- Property file location
- User-agent and content type information

8.4.2.1 Property files

Each transcoding JavaBean relies on the property files supplied with Transcoding Publisher to provide preference settings that indicate how a transformation should be performed for a particular device. Information about which XML stylesheets are registered is also included in the property files.

To define the location of the property files to the JavaBean, you may use the *setInstallpath()* and *setSystemDatabaseDirectory()* methods, as in the following example taken from the TransformTool.java source code tool:

```
imageTranscoderBean = new ImageTranscoderBean();
imageTranscoderBean.setInstallPath("./");
imageTranscoderBean.setSystemDatabaseDirectory("etc");
imageTranscoderBean.initialize();
```

Figure 97. TransformTool sample program

The TransformTool runs out of the Transcoding Publisher installation directory, so in this case, the shorthand approach of identifying the

installation directory (“.”) is sufficient. If your application resides in another path, be sure to specify the complete path for Transcoding Publisher (for example, `setInstallPath(“C:\\Program Files\\IBMTrans\\”)`). Because the property files are always installed in the same path relative to the installation directory, the use of `setSystemDatabaseDirectory(“etc”)` should always be appropriate, as long as the Transcoding Publisher directory is correctly specified.

8.4.2.2 User-agent and content type information

In addition to knowing where the property files reside, the transcoding JavaBeans also need to know the user-agent value associated with the target device and the MIME type of the content that is being passed to the JavaBean for transcoding. This information is stored in an `HttpPreferenceBundle` object, in a series of key-value pairs. Because the user-agent value and the content type are frequently used, special methods have been created to facilitate their setting, namely `setUserAgent()` and `setContentType()`.

The following sample statements show how these values can be set:

- `HttpPreferenceBundle http_pb = new HttpPreferenceBundle()`
- `html_pb.setUserAgent(“Mozilla/4.6”)`
- `html_pb.setContentType(“text/html”)`

Note that the user-agent value need not be specified in its entirety (such as `Mozilla/4.6 [en] (WinNT;U)`), but only enough of it to distinguish it from other user-agent values that are known to Transcoding Publisher.

After setting these values, you can invoke the transcoding operation with the `service()` method, passing in the `HttpPreferenceBundle` object you created along with the content:

```
transcodePage = htmlHandlerBean.service(html_pb, new
String(transcodeContentByteArray));
```

where `transcodeContentByteArray` contains the original data to be transcoded.

8.4.3 Using the JavaBeans with Linux

If you are using the `ImageTranscoderBean` on the Linux operating system, you will need to ensure that the path information for the Linux environment is updated to include `lib.linux`. The `RunTranscoding.sh` file provides an example of how you might test to see if `lib.linux` has been added to the path:

```
liblinuxpath=`echo $LD_LIBRARY_PATH | grep lib.linux`
if [ "$liblinuxpath" = "" ] ; then
export LD_LIBRARY_PATH=$PROD/lib.linux
fi
```

8.5 Scenario: invoking WTP JavaBeans from a servlet

In this section we present a sample scenario where we show you how a servlet invokes selected JavaBeans provided with WTP in order to transcode HTML and a gif image. Figure 98 illustrates the sample scenario. The following elements are used:

1. TxDemo1.jsp is a JSP that includes a form to request an option from the end user at the browser. This is a very simple input page and its form contains a radio button with the option to select whether WTP JavaBeans should be invoked or not. This page does not invoke the JavaBeans but the option is posted with the Submit button to the sample servlet.
2. TxDemo1.java is the sample servlet that will invoke WTP JavaBeans based on the selected option. The servlet is posted by the form in the input JSP.
3. ImageTranscoderBean is the JavaBean provided by WTP for image transcoding.
4. HtmlReducerBean is the JavaBean provided by WTP for HTML transcoding.

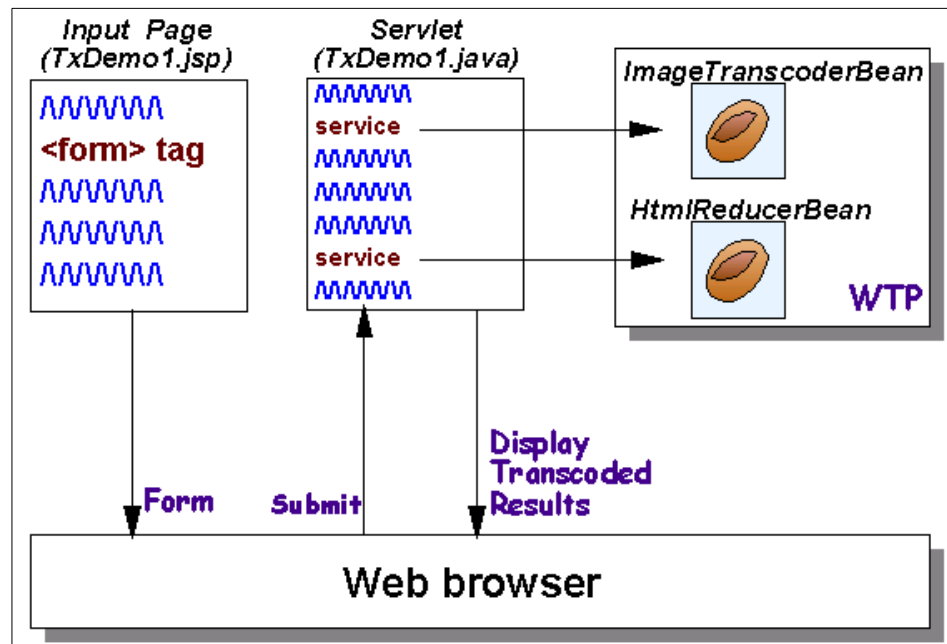


Figure 98. Scenario: invoking WTP JavaBeans from servlets

TxDemo1.jsp

The input page is shown in Figure 99. It contains the form with the radio button to select the option to transcode the sample image and HTML content.

```
<HTML>
<HEAD>
<TITLE>Hello World - JSP</TITLE>
</HEAD>

<BODY>
<BIG>This is a demo using Transcoding as JavaBeans. Today's date is:
<%= new java.util.Date() %>
</BIG>

<FORM METHOD="POST" ACTION="<%= response.encodeUrl("../servlet/TxDemo1") %>">
<P>Select Option and Submit
<P>
<INPUT TYPE="radio" NAME="txVar" VALUE = "Y" CHECKED>Transcode
<INPUT TYPE="radio" NAME="txVar" VALUE = "N" >Do not Transcode
<P>
<INPUT TYPE="submit" VALUE="Submit">
</FORM>

</BODY>
<% out.close(); %>
</HTML>
```

Figure 99. Sample JSP input page including a form

When the user selects the option to transcode (using the radio button) and he or she presses the **Submit** button, the sample servlet (TxDemo1.java) will be posted with the selected option.

You probably would like to copy this file (TxDemo1.jsp) to the Web server's HTML document root directory. For example:

C:\Program Files\IBM HTTP Server\htdocs

Then you can invoke this page from the browser, by specifying the proper URL. For example:

`http://<server-name>/TxDemo1.jsp`

TxDemo1.java

In this sample scenario, after the end user selects the option to transcode and clicks the **Submit** button, the TxDemo1.java program is posted as specified in the form tag in the TxDemo1.jsp page (see Figure 99).

The TxDemo1.java servlet is shown in Figure 100 on page 137 (part 1) and in Figure 101 on page 138 (part 2). It includes the statements to set the property file path, the user-agent and the content type information. It also includes the statements to instantiate the WTP JavaBean and invoke the transcoding function by calling the method with name service.

```

import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.ibm.transform.bean.*;

public class TxDemo1 extends HttpServlet {

    private String sTxVar;
    private String content;
    private byte[] origContentByteArray;
    private byte[] transcodedContentByteArray;
    private byte[] origImageByteArray;
    private byte[] transcodedImageByteArray;
    private HtmlReducerBean htmlReducerBean;
    private ImageTranscoderBean imageTranscoderBean;
    private InputStream is;

    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {

        res.setContentType("text/html");
        PrintWriter out = res.getWriter();

        content = new String("<HTML>\n" +
            "<HEAD>\n" +
            "<meta content= \"text/html; charset=iso-8859-1\">\n" +
            "<TITLE>Hello World Servlet</TITLE></HEAD>\n" +
            "<BODY LINK=\"#FFFFFF\" ULINK=\"#FFFFFF\" ALINK=\"#FFFFFF\" bgcolor=WHITE>\n");
        if(sTxVar.equalsIgnoreCase("y"))
            content = content + "<img src=\"myapps/images/masthead1.gif\" ALIGN=\"top\">\n";
        else
            content = content + "<img src=\"myapps/images/masthead.gif\" ALIGN=\"top\">\n";

        content = content + "<BIG><P>Hello World from Servlet</BIG>\n</BODY></HTML>";
        origContentByteArray = content.getBytes();

        if(sTxVar.equalsIgnoreCase("y"))
        {
            try {

                htmlReducerBean = new HtmlReducerBean();
                htmlReducerBean.setInstallPath("C:\\Program Files\\IBMTans\\");
                htmlReducerBean.setSystemDatabaseDirectory("etc");
                htmlReducerBean.initialize();

                String transcodePage;
                transcodedContentByteArray = (byte[]) origContentByteArray.clone();

                imageTranscoderBean = new ImageTranscoderBean();
                imageTranscoderBean.setInstallPath("C:\\Program Files\\IBMTans\\");
                imageTranscoderBean.setSystemDatabaseDirectory("etc");
                imageTranscoderBean.initialize();

                HttpPreferenceBundle html_pb = new HttpPreferenceBundle();
                html_pb.setUserAgent("MSIE 5.");
                html_pb.setContentType("text/html");
                transcodePage = htmlReducerBean.service(html_pb, new String(transcodedContentByteArray));
                transcodedContentByteArray = transcodePage.getBytes();
            } catch (Exception e)
            {
                out.println("<HTML>");
                out.println("<HEAD>");
                out.println("<meta content= \"text/html; charset=iso-8859-1\">");
                out.println("<TITLE>Error</TITLE></HEAD>");
                out.println("<BODY LINK=\"#FFFFFF\" ULINK=\"#FFFFFF\" ALINK=\"#FFFFFF\" bgcolor=WHITE>");
                out.println("<BIG><P>Error transcoding the html!</BIG>");
                out.println(e);
                out.println("</BODY></HTML>");
                return;
            }
        }
    }
}

```

Figure 100. Sample servlet (TxDemo1.java) invoking WTP JavaBeans - Part 1

```

try {
    File file = new File("c:\\WebSphere\\AppServer\\TxApps\\images\\masthead.gif");
    origImageByteArray = loadFileContent(file);
    if(origImageByteArray == null)
    {
        out.println("<HTML>");
        out.println("<HEAD>");
        out.println("<meta content='text/html; charset=iso-8859-1'>");
        out.println("<TITLE>Error</TITLE></HEAD>");
        out.println("<BODY LINK='\\' ULINK='\\' ALINK='\\' bgcolor=WHITE>");
        out.println("<BIG><P>Error opening the gif file!</BIG>");
        out.println("</BODY></HTML>");
        return;
    }
    transcodedImageByteArray = (byte[]) origImageByteArray.clone();
    HttpPreferenceBundle image_bundle = new HttpPreferenceBundle();
    image_bundle.setUserAgent("MSIE 5.");
    image_bundle.setContentType("image/gif");
    is = ImageTranscoderBean.Service(image_bundle, transcodedImageByteArray); ←
    int bytesRead;
    FileOutputStream outFile = new FileOutputStream(
        new File("c:\\WebSphere\\AppServer\\TxApps\\images\\masthead1.gif"));
    BufferedOutputStream buff = new BufferedOutputStream(outFile);
    while((bytesRead = is.read(transcodedImageByteArray)) >= 0)
        buff.write(transcodedImageByteArray, 0, bytesRead);
    buff.flush();
    buff.close();
} catch (Exception e)
{
    out.println("<HTML>");
    out.println("<HEAD>");
    out.println("<meta content='text/html; charset=iso-8859-1'>");
    out.println("<TITLE>Error</TITLE></HEAD>");
    out.println("<BODY LINK='\\' ULINK='\\' ALINK='\\' bgcolor=WHITE>");
    out.println("<BIG><P>Error transcoding the image!</BIG>");
    out.println(e);
    out.println("</BODY></HTML>");
    return;
}
out.print(new String(transcodedContentByteArray));
}
else
    out.print(new String(origContentByteArray));
}

public void doPost(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {

    sTxVar = req.getParameterValues("txVar")[0];
    doGet(req, res);
}

public byte[] loadFileContent(File file) {
    try {
        byte[] bytes = new byte[(int) (file.length())];
        FileInputStream fis = new FileInputStream(file);
        DataInputStream dis = new DataInputStream(fis);
        dis.readFully(bytes);

        return(bytes);
    } catch (java.io.FileNotFoundException e) {
        return null;
    } catch (java.io.IOException e) {
        return null;
    }
}
}

```

Figure 101. Sample servlet (TxDemo1.java) invoking WTP JavaBeans - Part 2

This servlet first creates the HTML code and saves it into a string (content). This HTML code will be transcoded or not depending on the selected option. The image that is called inside the HTML code should also be transcoded or not. If the user chooses the option **Do not Transcode**, the image to be called should be the original image "masthead.gif". However, if the user chooses the

option **Transcode**, a new image should be created, the transcoded image name will be “masthead1.gif”.

```
content = new String("<HTML>\n" +
    "<HEAD>\n" +
    "<meta content = \"text/html; charset=iso-8859-1\">\n" +
    "<TITLE>Hello World Servlet</TITLE></HEAD>\n" +
    "<BODY LINK=\"#FFFFFF\" ULINK=\"#FFFFFF\" ALINK=\"#FFFFFF\" bgcolor=WHITE>\n");
if(sTxVar.equalsIgnoreCase("y"))
    content = content + "<img src=\"../myapps/images/masthead1.gif\" ALIGN=\"top\">\n";
else
    content = content + "<img src=\"../myapps/images/masthead.gif\" ALIGN=\"top\">\n";
content = content + "<BIG><P>Hello World from Servlet</BIG>\n</BODY></HTML>";
origContentByteArray = content.getBytes();
```

Figure 102. Creating HTML code

In this sample scenario, if the user selects the option not to transcode, all the servlet does is print the HTML code as is, without any changes. However, if the end user selects the option to transcode, the servlet calls the **ImageTranscoderBean** and the **HtmlReducerBean** JavaBeans, for the image transcoding and the HTML transcoding respectively.

As the application resides in a path other than the install path, the complete path was specified in the setInstallPath method. The JavaBean will look for the appropriate profiles in the directory “etc” from the install path (see Figure 103).

```
htmlReducerBean = new HtmlReducerBean();
htmlReducerBean.setInstallPath("C:\\Program Files\\IBMTrans\\");
htmlReducerBean.setSystemDatabaseDirectory("etc");
htmlReducerBean.initialize();

String transcodePage;
transcodedContentByteArray = (byte[]) origContentByteArray.clone();

imageTranscoderBean = new ImageTranscoderBean();
imageTranscoderBean.setInstallPath("C:\\Program Files\\IBMTrans\\");
imageTranscoderBean.setSystemDatabaseDirectory("etc");
imageTranscoderBean.initialize();
```

Figure 103. Specifying the install path

In order to transcode, the transcoding JavaBeans need to know the user-agent value associated with the target device and the MIME type of the content that is being passed. That can be done with the methods setUserAgent and setContentType as illustrated in Figure 104 on page 140.

The WTP JavaBean method to actually execute the transcoding process is the service method. For example, in Figure 104 on page 140, the service method is invoked for the object htmlReducerBean which is an instance of the HtmlReducerBean JavaBean.

```
HttpPreferenceBundle html_pb = new HttpPreferenceBundle();
html_pb.setUserAgent("MSIE 5.");
html_pb.setContentType("text/html");
transcodePage = HTMLReducerBean.service(html_pb, new String(transcodedContentByteArray));
transcodedContentByteArray = transcodePage.getBytes();
```

Figure 104. Specifying the user-agent and the MIME type

There are several ways you can run servlets in a WAS environment. In this section we describe a simple way to do this but it is not necessarily the best and you should consult your WAS administrator for the procedure used in your installation.

Once the servlet is compiled, you may want to execute the following steps to run the sample servlet:

1. Copy the compiled file (class) to the servlet root directory, for example:
C:\WebSphere\AppServer\servlets
2. Create the directory TxApps\images below the directory
WebSphere\AppServer.
3. Create an alias for this directory:
 - a. Open the httpd.conf configuration file in the IBM HTTP Web server directory:
<Install_Path>\conf
For example: C:\Program Files\IBM HTTP Server\conf\httpd.conf
 - b. Include this alias along with other alias definitions previously included by WAS for example:
Alias /myapps/ "C:/WEBSPH~1/APPSER~1/TxApps/"
 - c. Save the configuration file and refresh the IBM HTTP server to pick up the new alias.
4. Call the TxDemo1.jsp input page from your browser as illustrated in Figure 105 on page 141.

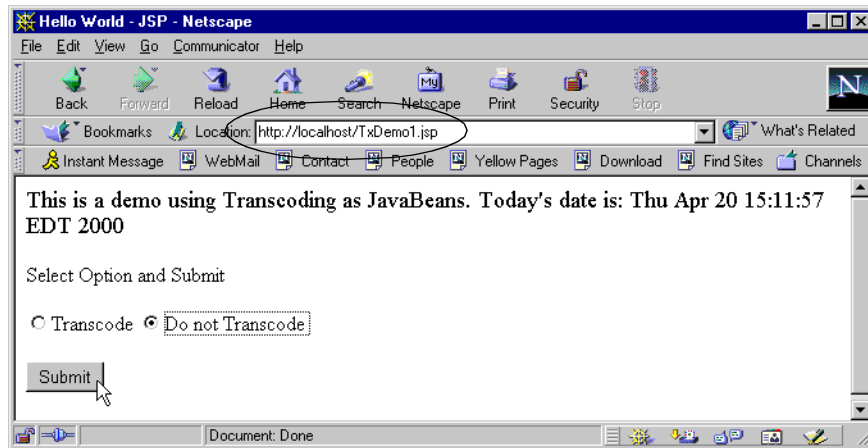


Figure 105. Input page - selecting Do not Transcode option

When you select the do not transcode option the servlet is posted but it will not invoke the WTP JavaBeans so no transcoding takes place. Figure 106 shows the original page.

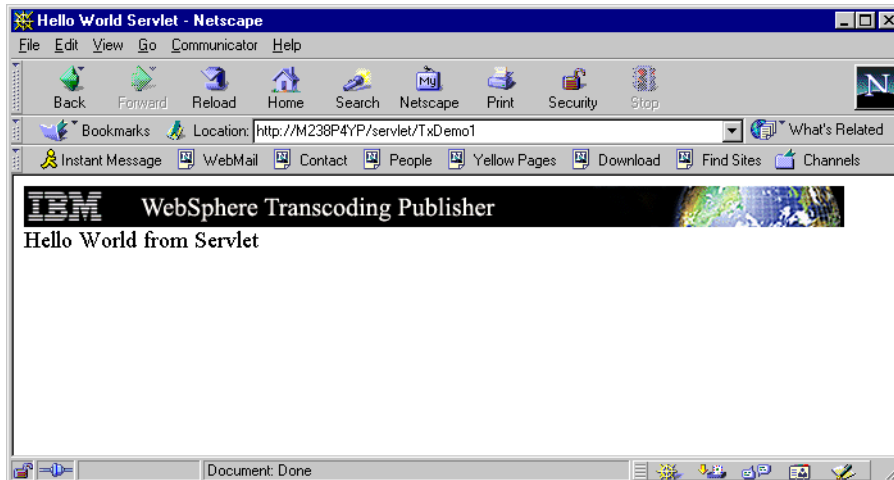


Figure 106. Original page (not transcoded) created by the sample servlet

Next, we select the option to transcode the servlet content (gif image and HTML). The servlet invokes the WTP JavaBeans to accomplish this function. Figure 107 on page 142 shows the input page (TxDemo1.jsp) with the option selected. Click the **Submit** button to post the sample servlet.

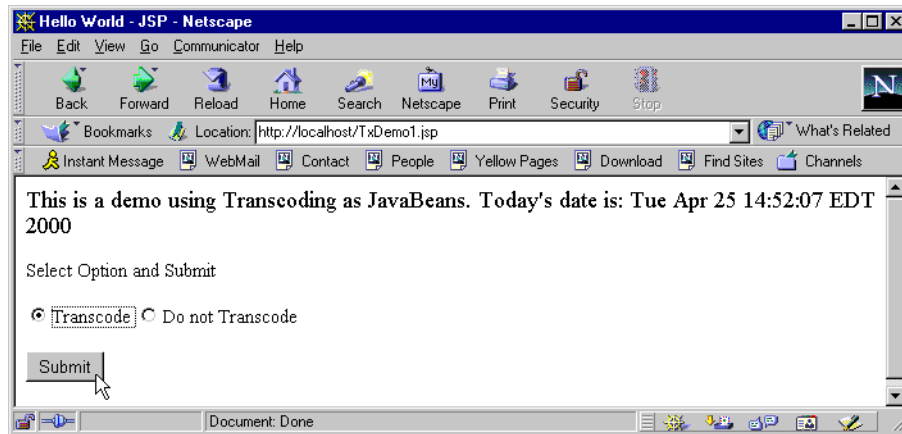


Figure 107. Selecting the option to Transcode the servlet content

Figure 108 illustrates the transcoded page. For example, it shows image reduction as configured in the device profile for this client type (user-agent was set to “MSIE 5.” in the sample servlet (see Figure 100 on page 137).

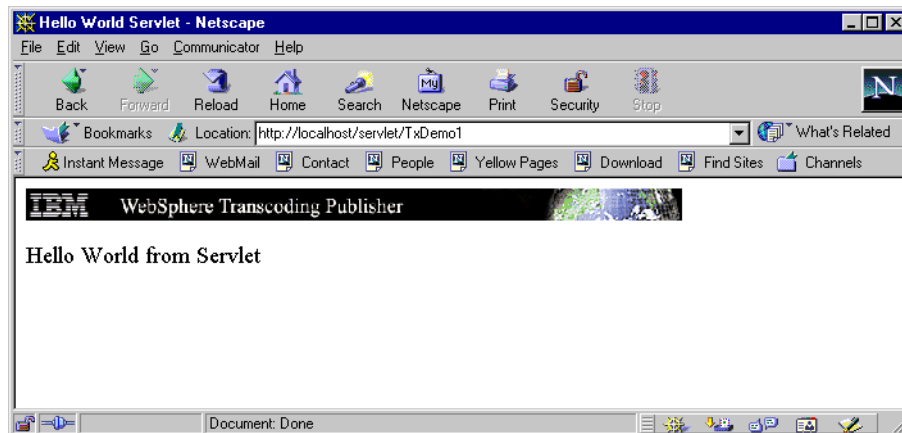


Figure 108. Transcoded page showing image reduction

8.6 Scenario: invoking WTP JavaBeans from JSPs

In this scenario we show you how to invoke WTP JavaBeans from JavaServer Pages (JSP). Figure 109 on page 143 illustrates this scenario. The WTP ImageTranscoderBean is invoked from a sample JavaServer Page (JSP) to transcode an image file.

The following elements are used:

1. TxDemo2.jsp is a JSP that includes a form to request an option from the end user at the browser. This is a simple input page and its form contains a radio button with the option to select the target device. This page does not invoke the WTP JavaBean but merely collects the option to post the sample JavaServer Page (TxDemo3.jsp) when the Submit button is pressed.
2. TxDemo3.jsp is the sample JavaServer Page (JSP) based on the sample ImageTranscoder bean that will invoke the WTP JavaBean passing the selected option (target device). This JSP is posted by the form in the input JSP (TxDemo2). You should be aware that eventually WAS will compile this JSP into a servlet for execution.
3. ImageTranscoderBean is the JavaBean provided by WTP for image transcoding.

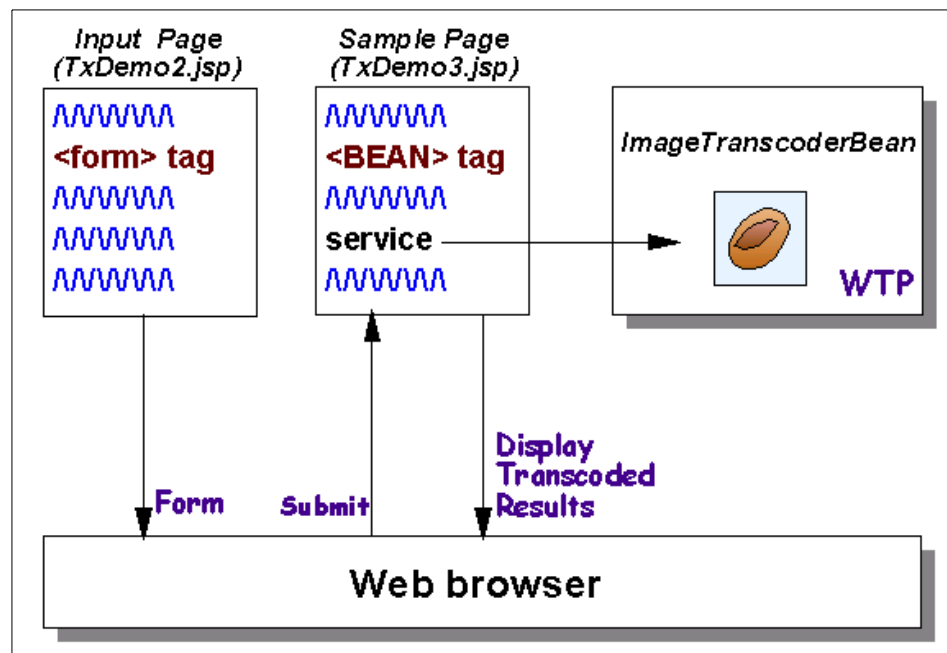


Figure 109. Scenario: invoking WTP JavaBeans from a JSP

TxDemo2.jsp

The input page is shown in Figure 110 on page 144. It contains the form with the radio button to select the target device (IE5, Windows CE, Palm Pilot or WAP phone) to transcode the sample image.

When the user selects the target device (using the radio button) and he or she presses the **Submit** button, the sample JSP (TxDemo3.jsp) will be posted with the selected option as a parameter.

```
<HTML>
<TITLE>TxDemo2</TITLE>
<BODY BGCOLOR="white">
<H1 align="center">WebSphere Transcoding Publisher</H1>
<FORM METHOD=GET ACTION=TxDemo3.jsp>
<IMG SRC="/myapps/images/ibmtrans.jpg" HSPACE=100></IMG>
<%
    String device_selected = null;
    if (device_selected == null) device_selected = "IE";
%>
<PRE>
<H3 align="center">Please select the target device to transcode the image</H3>
    <INPUT TYPE=RADIO NAME=device_selected VALUE=IE CHECKED>Internet Explorer 5
    <INPUT TYPE=RADIO NAME=device_selected VALUE=WinCE UNCHECKED>Hand Held Win CE
        <INPUT TYPE=SUBMIT VALUE="Transcode">
    <INPUT TYPE=RADIO NAME=device_selected VALUE=Paln UNCHECKED>Palm Pilot Hand Web
    <INPUT TYPE=RADIO NAME=device_selected VALUE=WAP UNCHECKED>WML WAP Phone
</PRE>
</FORM>
</BODY>
</HTML>
```

Figure 110. Input page (TxDemo2.jsp)

TxDemo3.jsp

This is the sample JSP that invokes the WTP JavaBean. It uses the <BEAN> tag to define the ImageTranscoderBean JavaBean. It also executes java code to set the user-agent (setUserAgent method) and invokes the transcoding function by executing the service method.

The sample JavaServer Page (TxDemo3.jsp) is shown in Figure 111 on page 145 and in Figure 112 on page 146.

Copy the two JavaServer Pages (input page and sample page) to the Web server's HTML document root directory. Check with your WAS administrator for the correct folder for JSPs in your installation. For example:

C:\Program Files\IBM HTTP Server\htdocs

```

<HTML>
<TITLE>TxDemo3</TITLE>
<BODY BGCOLOR="white">
<H1 align="center">WebSphere Transcoding Publisher</H1>
</BR>
<%@ language = "java" %>
<%@ import = "java.io.*,com.ibm.wbi.util.*,com.ibm.transform.bean.*"%>

<script runat=server>

byte[] bytes;
InputStream is;
File file;
FileInputStream fis;
DataInputStream dis;
FileOutputStream fileos;
int bytesRead;
BufferedOutputStream buff;
HttpPreferenceBundle itb_pb;
String device;
String[] result;
String device_selected = null;
String success_statement = null;
</script>
<BEAN NAME="itb" TYPE="com.ibm.transform.bean.ImageTranscoderBean" INTROSPECT=NO SCOPE=REQUEST>
</BEAN>
<font size=4>
<ul>
<%itb.setInstallPath("c:" + File.separator + "Program Files" + File.separator + "IBMTrans"); %>
<%itb.setSystemDatabaseDirectory("etc");%>
</br>
<%
response.setHeader("Cache-Control", "no-cache");
result = request.getParameterValues("device_selected");
if (result != null) device=result[0];
try{
file = new File("c:\\WebSphere\\AppServer\\TxApps\\images\\ibmtrans.jpg");
bytes = new byte[(int) (file.length())];
fis = new FileInputStream(file);
dis = new DataInputStream(fis);
dis.readFully(bytes);
} catch (Exception e) {
e.toString();
}
%>
<%
itb_pb = new HttpPreferenceBundle();
if (device.equals("IE"))
{
success_statement="The image was successfully transcoded for an Internet Explorer device!";
itb_pb.setUserAgent("MSIE 5.");
}
if (device.equals("WinCE"))
{
success_statement="The image was successfully transcoded for a Hand Held WinCE device!";
itb_pb.setUserAgent("Windows CE");
}
if (device.equals("Palm"))
{
success_statement="The image was successfully transcoded for a Palm device!";
itb_pb.setUserAgent("Mozilla/3.0 (compatible; HandHTTP 1.1");
}
if (device.equals("WAP"))
{
success_statement="The image was successfully transcoded for a WML WAP Phone device!";
itb_pb.setUserAgent("Nokia-WAP-Toolkit/1.2");
}
itb_pb.setContentType("image/jpeg");

```

Figure 111. Sample JSP (TxDemo3.jsp) invoking WTP JavaBean (Part 1)

```

try {
    is = itb.service(itb_pb, bytes);
} catch (Exception e) {
    e.toString();
}

try{
    fileos = new FileOutputStream(new File("c:\\WebSphere\\AppServer\\TxApps\\images\\newImage.jpg"));
    buff = new BufferedOutputStream(fileos);
    while ((bytesRead = is.read(bytes)) >= 0)
        buff.write(bytes, 0, bytesRead);
    buff.flush();
    buff.close();
} catch (Exception e) {
    e.toString();
}
}
%>
<PRE><H3>
    <%=success_statement%>
</H3></PRE>
</BR>
<IMG SRC="/myapps/images/newImage.jpg" HSPACE=80></IMG>
</u1>
</font>
</BODY>
</HTML>

```

Figure 112. Sample JSP (TxDemo3.jsp) invoking WTP JavaBean (Part 2)

Once the JavaServer Pages are available to WebSphere Application Server (WAS), we can now try the sample JavaServer Page. We suggest the following steps:

1. From a browser, enter the URL to invoke the input JSP containing the form to select the target device for example (see Figure 113 on page 147):
<http://<server-name>/TxDemo2.jsp>
2. Select the target device.
3. Click **Submit** to invoke the sample JSP. This action will post TxDemo3.jsp.
4. The sample JSP is executed (service method) and it invokes the WTP JavaBean with the proper user-agent value as illustrated in Figure 111 on page 145.

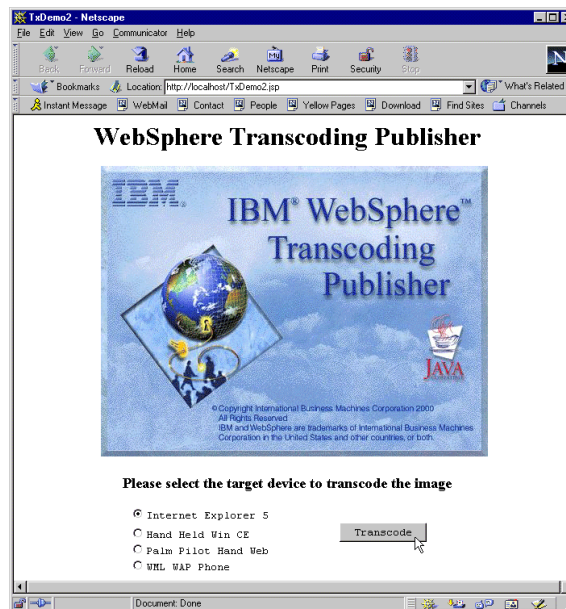


Figure 113. Invoking the TxDemo2.jsp input page from a browser

Figure 114 shows the transcoded image for an IE5 device.

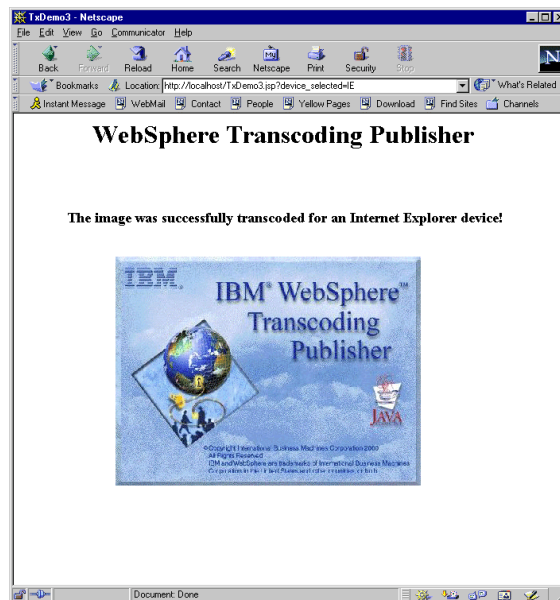


Figure 114. Transcoded image for IE5 device

Figure 115 shows the transcoded image for a Palm device.

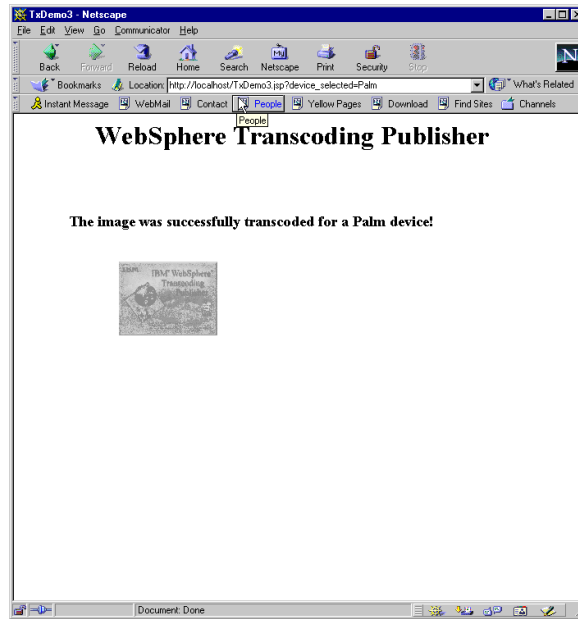


Figure 115. Transcoded image for Palm device

8.7 Scenario: invoking a WTP JavaBean to transcode XML

In this scenario we transcode XML to whatever the target device supports. For example, if the target device is a desktop browser, XML is transcoded to HTML. The sample servlet we provide for this scenario (TxDemoJsp.java) is based on the sample tool TranscodingSamples.

The TranscodingSamples is a sample servlet program provided by IBM WebSphere Transcoding Publisher Version 1.1 in the toolkit subdirectory. For more information about this tool see 7.5.1, “Sample TranscodingSamples” on page 114 and Appendix A, “TranscodingSamples sample program” on page 301.

In this scenario, the following elements are used:

1. TxDemoJsp.jsp is a JSP that includes a form to request an option from the end user at the browser. This is a simple input page and its form contains a list to select the XML file to transcode and a radio button to select the target device. This page does not invoke the JavaBean but the options are posted with the submit button to the sample servlet.

2. TxDemoJsp.java is the sample servlet that will invoke the WTP JavaBean (XmlHandlerBean). The servlet is posted by the form in the input JSP.
3. XmlHandlerBean is the JavaBean provided by WTP for XML transcoding.

The scenario is illustrated in Figure 116.

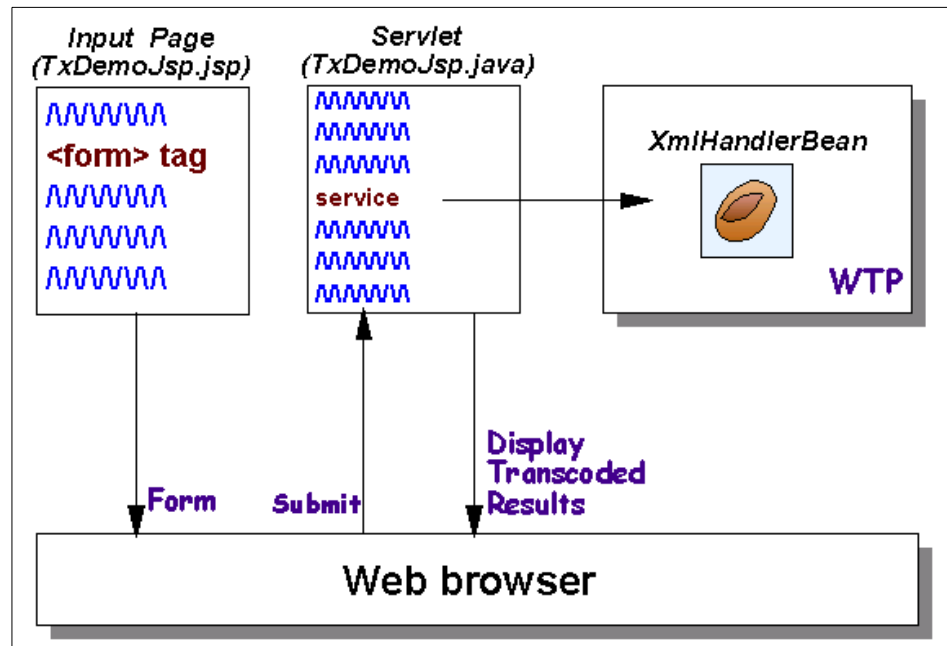


Figure 116. Sample scenario: invoking WTP XMLHandlerBean from servlet

The input page is shown in Figure 117 on page 150. You will need to make the input page (TxDemoJsp.jsp) available to WebSphere Application Server. Check with your WAS administrator for the proper folder where this page is supposed to reside. For example, copy this file to the Web server's HTML document root directory:

C:\Program Files\IBM HTTP Server\htdocs

To invoke this page from a browser, use the required URL. For example:

`http://<server-name>/TxDemoJsp.jsp`

```

<HTML>
<HEAD>
<TITLE>FlightInfo - JSP</TITLE>
</HEAD>

<BODY>
<BIG>This is a demo using Transcoding as JavaBeans. Today's date is:
<%= new java.util.Date() %>
</BIG>

<FORM METHOD="POST" ACTION="<%= response.encodeUrl("../servlet/TxDemoJsp") %>">
<P><H4>Select Option:</H4>
<INPUT TYPE="radio" NAME="txVar" VALUE="Y" CHECKED>Transcode
<INPUT TYPE="radio" NAME="txVar" VALUE="N" >Do not Transcode</BR></BR>
<P><H4>Enter the xml file:</H4>
<INPUT TYPE="text" NAME="name" VALUE="FlightInfo.xml"</BR>
<PRE>
<P><H4>Select device and submit</H4>
<SELECT NAME="type" MULTIPLE>
<OPTION SELECTED>IE</OPTION>
<OPTION>WinCE</OPTION>
<OPTION>Palm</OPTION>
<OPTION>WAP</OPTION>
</SELECT>      <INPUT TYPE="submit" VALUE="Submit"></BR></BR>
</PRE>
</FORM>
</BODY>
<% out.close(); %>
</HTML>

```

Figure 117. Sample input page

Figure 118 shows the input page including the form.

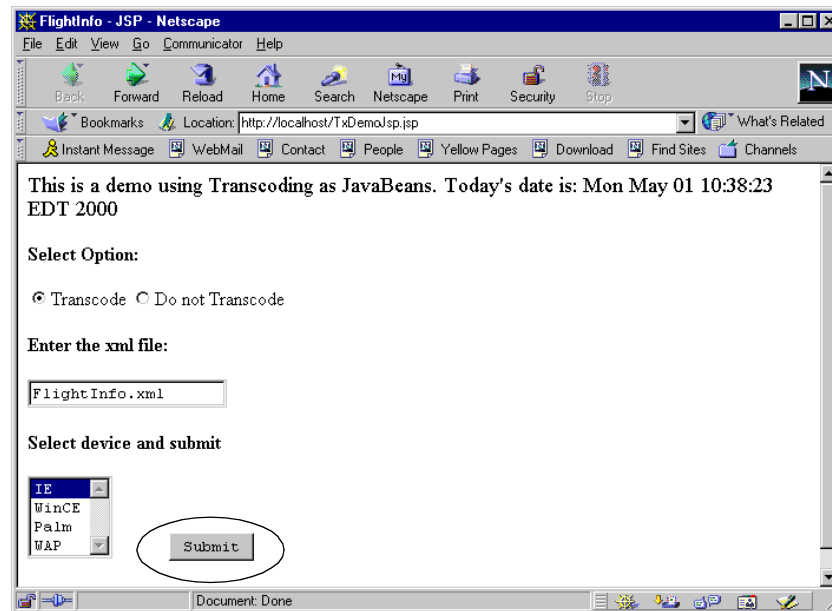


Figure 118. Requesting input page

Once the option to transcode, the XML input file, and the target device have been selected, click the **Submit** button to post the sample servlet.

The servlet is shown in Figure 119 (part 1) and Figure 120 on page 152 (part 2).

```
import java.io.*;
import java.text.*;
import java.util.*;
import java.util.Properties;
import javax.servlet.*;
import javax.servlet.http.*;
import com.ibm.transform.bean.*;

public class TxDemoJsp extends HttpServlet {

    private String content;
    private byte[] origContentByteArray;
    private byte[] transcodedContentByteArray;
    private XmlHandlerBean xmlHandlerBean;
    private final static String TOOLKIT_HOME = "toolkit";
    private final static String REQUESTABLE_SAMPLES = TOOLKIT_HOME + File.separator + "RequestableSamples.prop";
    private String installPath = ".";
    private Properties samples = null;
    private String name;
    private String sTxVar;
    private String type;

    public void init(ServletConfig config) throws ServletException
    {
        super.init(config);

        String temp = getInitParameter("InstallPath");
        if(temp != null)
            installPath = temp;

        String tableName = installPath + File.separator + REQUESTABLE_SAMPLES;
        samples = new Properties();
        try {
            samples.load(new FileInputStream(new File(tableName)));
        } catch (Exception e)
        {
            getServletContext().log("TxDemoJsp: Unable to load " + tableName);
        }
    }

    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {

        PrintWriter out = res.getWriter();

        if(name == null)
        {
            res.sendError(HttpServletResponse.SC_BAD_REQUEST);
            return;
        }
        String file = samples.getProperty(name);
        if(file == null)
        {
            res.sendError(HttpServletResponse.SC_FORBIDDEN);
            return;
        }
        String document = installPath + File.separator + TOOLKIT_HOME + File.separator + file.replace('/', File.separatorChar);
        String contentType = getServletContext().getMimeType(document);
        if(contentType == null)
            contentType = "text/plain";
        contentType = contentType.trim(); // Required on some versions of WAS
        res.setContentType(contentType);

        File NewFile = new File(document);
        origContentByteArray = loadFileContent(NewFile);
        if(origContentByteArray == null)
        {
            out.println("Error loading the file!");
            return;
        }
    }
}
```

Figure 119. Sample servlet (TxDemoJsp.java) - Part 1

```

if(sTxVar.equalsIgnoreCase("y"))
{
    try {

        xmlHandlerBean = new XmlHandlerBean();
        xmlHandlerBean.setInstallPath(installPath);
        xmlHandlerBean.setSystemDatabaseDirectory("etc");
        xmlHandlerBean.initialize();

        String transcodePage;
        transcodedContentByteArray = (byte[]) origContentByteArray.clone();

        HttpPreferenceBundle xml_pb = new HttpPreferenceBundle();
        if(type.equalsIgnoreCase("IE"))
            xml_pb.setUserAgent("MSIE 5.");
        else if(type.equalsIgnoreCase("WinCE"))
            xml_pb.setUserAgent("Windows CE");
        else if(type.equalsIgnoreCase("Palm"))
            xml_pb.setUserAgent("Mozilla/3.0");
        else
            xml_pb.setUserAgent("Nokia-WAP-Toolkit/1.2");
        xml_pb.setContentType(contentType);
        transcodePage = xmlHandlerBean.service(xml_pb, new String(transcodedContentByteArray));
        transcodedContentByteArray = transcodePage.getBytes();
    } catch (Exception e)
    {
        out.println("Error transcoding!");
        return;
    }

    out.print(new String(transcodedContentByteArray));
}
else
    out.print(new String(origContentByteArray));
}

public void doPost(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {

    name = req.getParameterValues("name")[0];
    sTxVar = req.getParameterValues("txVar")[0];
    type = req.getParameterValues("type")[0];
    doGet(req, res);
}

public byte[] loadFileContent(File file) {
    try {

        byte[] bytes = new byte[(int) (file.length())];
        FileInputStream fis = new FileInputStream(file);
        DataInputStream dis = new DataInputStream(fis);
        dis.readFully(bytes);

        return(bytes);
    } catch (java.io.FileNotFoundException e) {
        return null;
    } catch (java.io.IOException e) {
        return null;
    }
}
}

```

Figure 120. Sample servlet (TxDemoJsp.java) - Part 2

Same as in the TranscodingSamples servlet, this servlet looks for a file called RequestableSamples that lists the files to be transcoded (this mechanism provides some type of security). In this case, only FlightInfo.xml is allowed for transcoding. The FlightInfo.xml file contains the XML code and it is listed in Figure 87 on page 115.

Once the servlet is compiled, you can follow these steps to have it available in WebSphere Application Server (WAS):

1. Copy the class file to the servlet root directory for example:
C:\WebSphere\AppServer\servlets
2. Define TxDemoJsp servlet in WAS (configuration). For details on how to define a servlet in WAS see 7.5.2.1, “Suggested procedure” on page 116.
3. Register FlightInfoForNet_IE.xsl. See Figure 89 on page 118 for details on how to register a stylesheet. See also 7.3.1.1, “Stylesheet selection criteria” on page 104 to make sure the proper stylesheet will be selected for the target device you selected.
4. Request the JSP input page using the proper URL for example:
`http://<server-name>TxDemoJsp.jsp`
5. Select the options to transcode, enter the XML file name (FlightInfo.xml) and select the target device (see Figure 118 on page 150).

Figure 121 shows the transcoded results (XML-to-HTML) when the target device selected is a desktop (IE5).

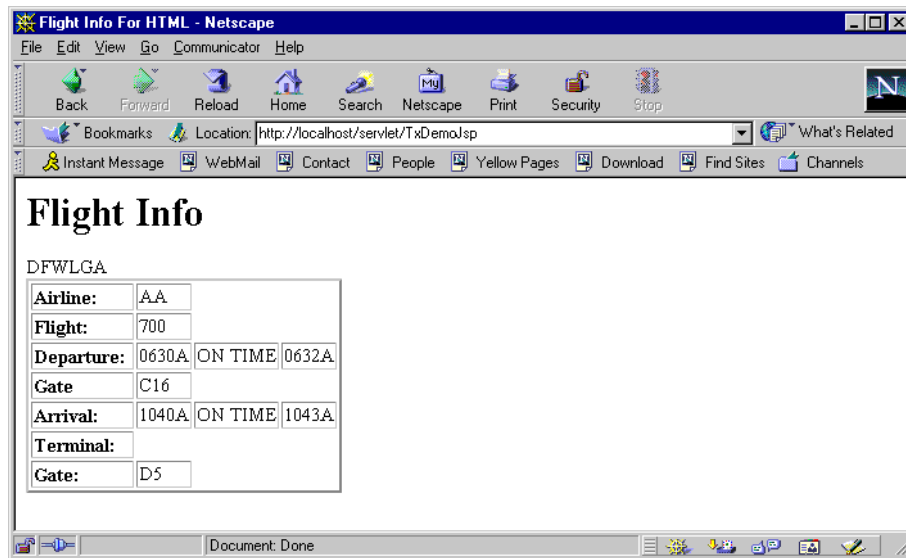


Figure 121. Transcoded XML to HTML results for the IE5 device

Figure 122 on page 155 shows the transcoded results (XML-to-HTML) when the target device selected is Windows CE.

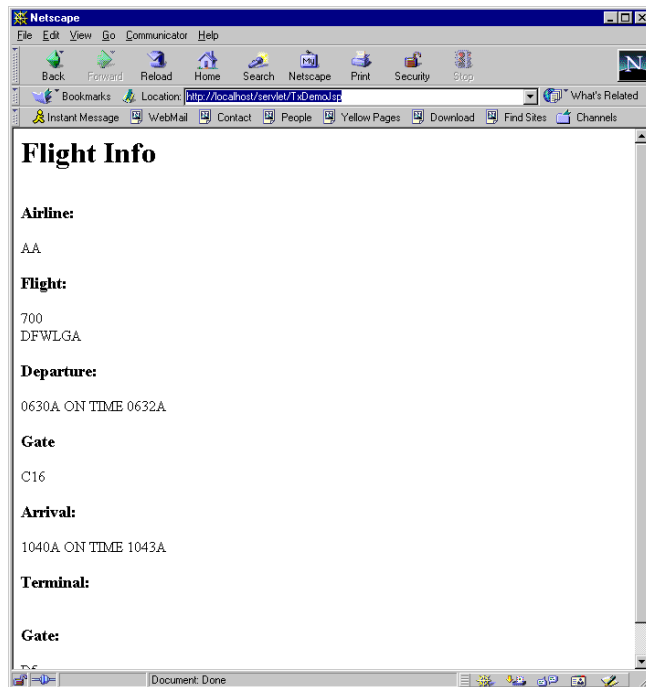


Figure 122. Transcoded XML to HTML results for Windows CE

Chapter 9. Using the Toolkit

The Developer's Toolkit is intended for device manufacturers, ISVs, and service people who want to extend the existing behavior of IBM WebSphere Transcoding Publisher by using the provided tools and APIs.

In this chapter we describe the tools that are the components of the toolkit. These tools help you use and develop extensions to Transcoding Publisher. In this chapter you will find information about the following tools:

- Transform Tool
- Request Viewer
- Creating preference profiles
- Snoop Tool
- Creating new transcoders

9.1 The Transform Tool

The Transform Tool has a dual role in Transcoding Publisher: as a programming sample of how to use the transcoding JavaBeans in an application and as a developer's tool for previewing the effect of various transcoding operations. By placing original content and transcoded content side by side, the Transform Tool demonstrates how a particular image or document will be affected by Transcoding Publisher's current settings.



Figure 123. The Transform Tool showing a transcoded image

For example, Figure 123 shows how an original image (in the left pane) would appear after being transcoded (in the right pane) and displayed on a Palm Pilot device running the HandWeb browser.

The Transform Tool bases its transcoding operations on the settings specified in the Transcoding Publisher preference profiles.

9.1.1 Starting the Transform Tool

To start the Transform Tool in Windows NT, do the following:

1. At the command prompt, change directory to the Transcoding Publisher installation directory, for example:

C:\Program Files\IBMTrans

2. Enter `TransformTool` at the prompt.

Alternatively, users also have the option of starting the tool by selecting:

Start->Programs->IBM Transcoding Publisher->Toolkit->Transform Tool

from the desktop as shown in Figure 124.

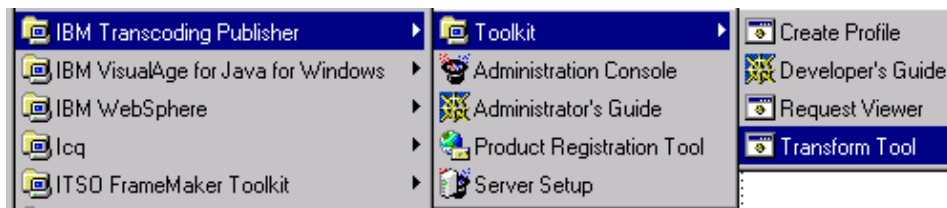


Figure 124. Starting the Transform Tool

9.1.2 Using the Transform Tool

To perform transcoding operations with the Transform Tool, you can do the following:

1. Select the content to be transcoded by choosing an option from the File pull-down menu. The Transform Tool enables you to choose from three kinds of data (see Figure 125 on page 159):
 - Load Image. It enables you to select a GIF or JPEG image.
 - Load HTML. It enables you to select an HTML document.
 - Load XML. It enables you to select an XML document.

When you have selected the file, its contents will be displayed in the left pane. Although images are rendered as such in the Transform Tool, HTML and XML documents are displayed in their source format, rather than as they might be formatted by a browser.

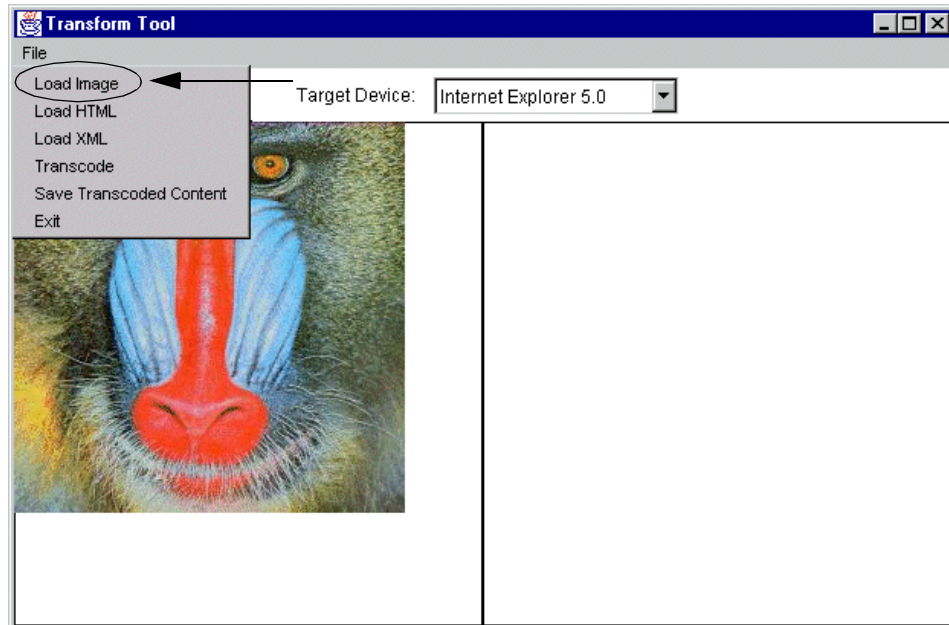


Figure 125. Image loaded in the Transform Tool

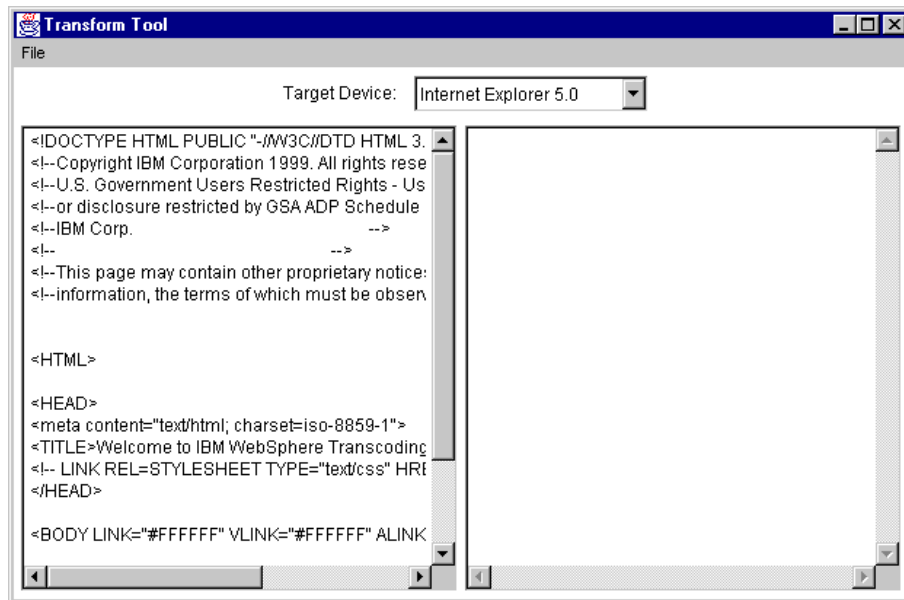


Figure 126. HTML loaded in Transform Tool

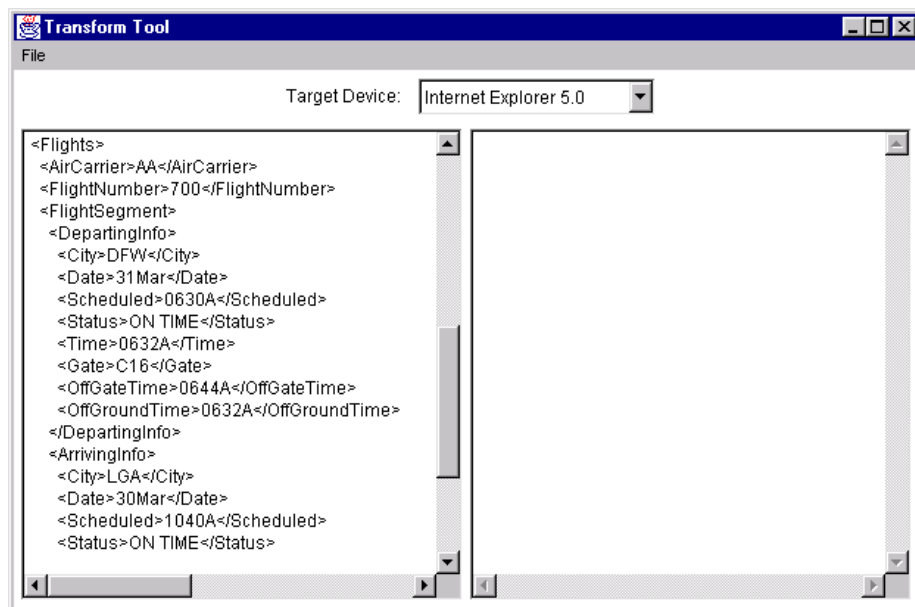


Figure 127. XML loaded in Transform Tool

2. Select a **Target Device** from the drop-down box. Currently, the tool supports the following values:
 - *Internet Explorer 5.0*
 - *Hand Held Windows CE*
 - *Palm Pilot HandWeb*
 - *WML WAP Phone*
3. Select **File->Transcode** to start the transcoding operation. The transcoded data will be displayed in the right pane. As with the untranscoded content, images are rendered appropriately, and HTML and XML output is shown in its source format.

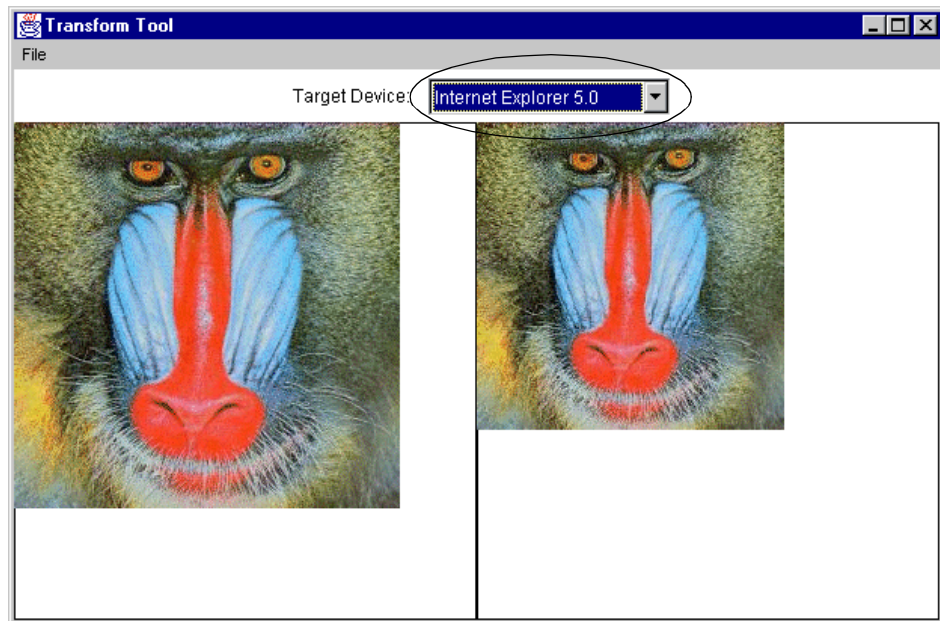


Figure 128. Image transcoded for Internet Explorer 5.0 with image reduction

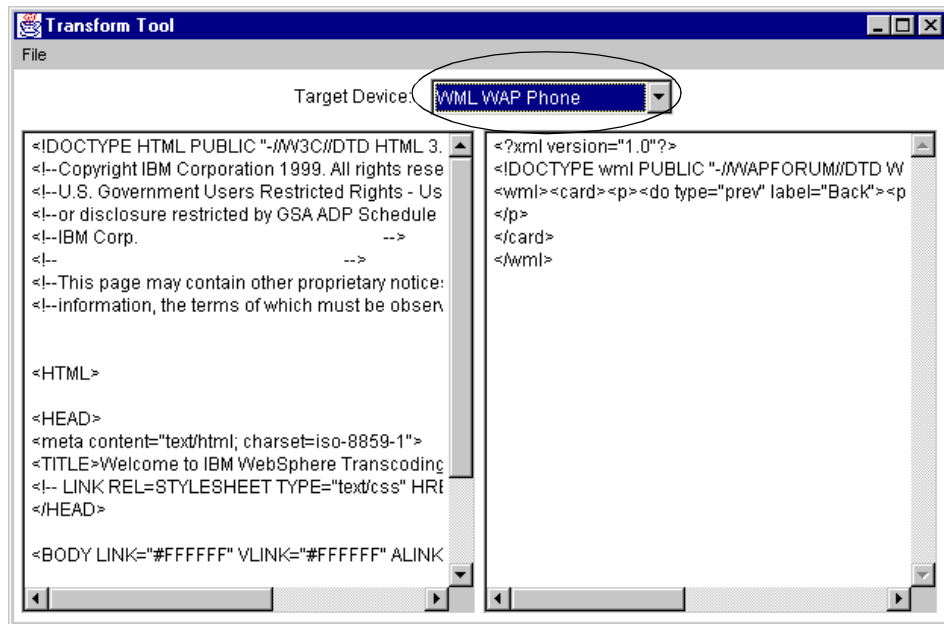


Figure 129. HTML transcoded for WML WAP phone

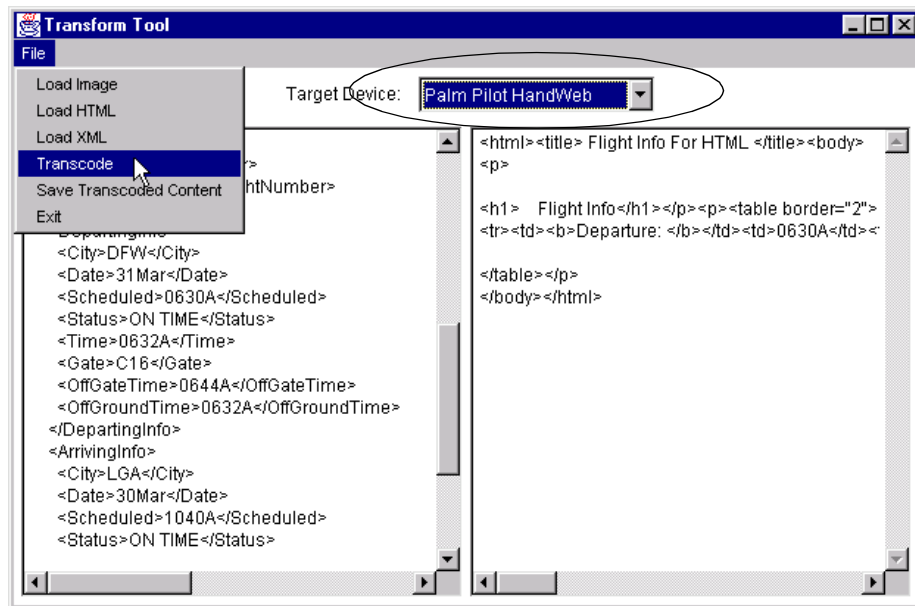


Figure 130. XML transcoded for Palm Pilot HandWeb

4. To keep the transcoded output for later use, select:
File->Save->Transcoded Content.
5. Select **File->Exit** to close the tool.

9.1.3 Using XML stylesheets with the Transform Tool

Transcoding Publisher includes several sample XML stylesheets you can use with the Transform Tool to observe how data can be transcoded from XML to HTML and other varieties of XML, such as WML. However, when using the Transform Tool to convert XML data through the use of stylesheets, you must ensure that each required stylesheet has been registered with Transcoding Publisher **prior to starting** the Transform Tool.

To register the XML stylesheets, do the following:

1. Start the Administration Console.
2. For each XML stylesheet you want to register, select a folder in which to store the stylesheet (such as the XML Stylesheet Selectors folder) and then use the **Register->XML Stylesheet** option to start the registration wizard. More detailed instructions for using the console are available from the online help.

The following information should be specified in the appropriate places in the wizard:

- **Location:** The sample files are located in the toolkit\stylesheets\samples directory under the Transcoding Publisher installation directory. There is a sample file for each device type supported by the Transform Tool.
- **Output content type:** Specify text/html for each stylesheet, with the exception of the one for WML, which content type should be text/vnd.wap.wml.
- **Input DTD:** Specify `Flights` for each stylesheet.

For more detailed instructions about using a Stylesheet see 7.5, “Sample Scenario: transcoding XML content” on page 113.

3. Update the transcoding server by selecting **File->Refresh Server.**
4. Exit the Administration Console and start the Transform Tool. A sample XML source file is also provided in
toolkit\stylesheets\samples\FlightInfo.xml.

Because the tool implements the transcoding JavaBeans model, it does not have a mechanism to detect dynamic updates from Transcoding Publisher. If

you register a new stylesheet while the Transform Tool is already running, stop and restart the tool to be sure that it has the current preference setting information from the transcoding server before attempting to perform any transcoding with the new stylesheet.

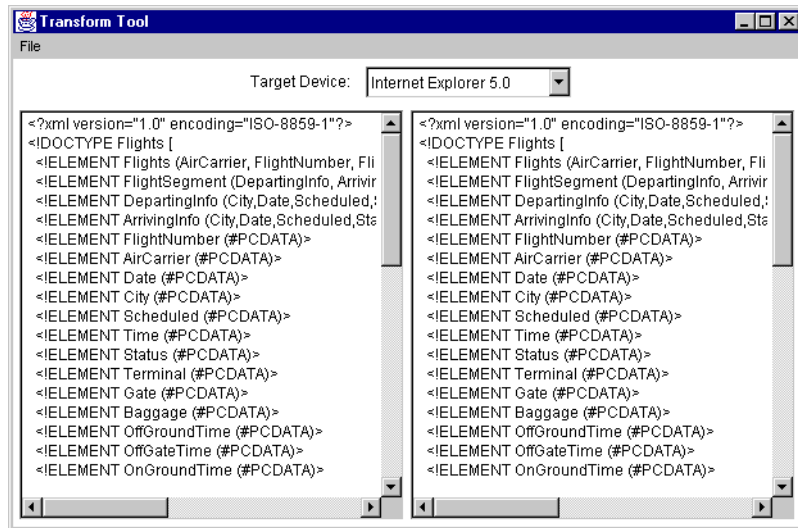


Figure 131. Calling the FlightInfo.xml without registering the stylesheet

Note

When you try to transcode an XML file without registering the stylesheet it will not be transcoded. After the stylesheet is registered the transcoding works properly.

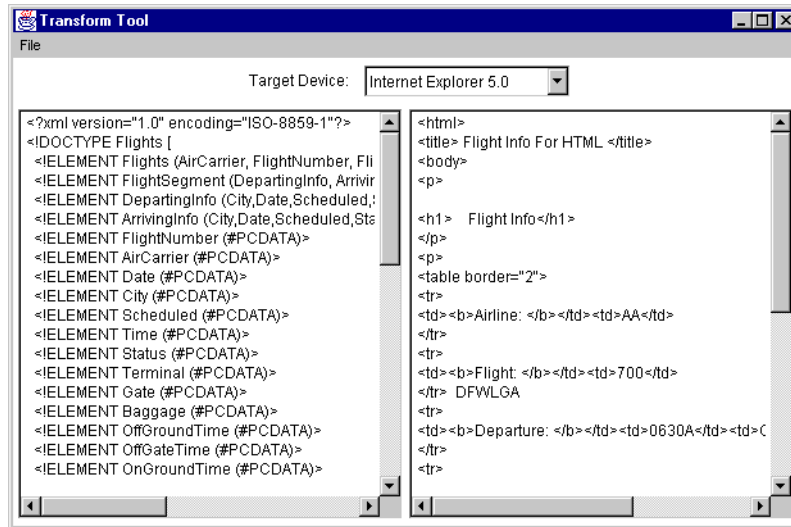


Figure 132. Calling the FlightInfo.xml after registering the stylesheet

9.1.4 The Transform Tool application

The Transform Tool source code is in the WTP directory:

```
<install_path>\toolkit\beansamples\TransformTool
```

This application uses transcoding beans to provide a stand-alone transcoded page/image generator and previewer. It has a graphical user interface tool which illustrates how to use the transcoding beans provided by Transcoding Publisher.

There are two values that should be provided for the user: The **Key Value Pair User-Agent/Device** and the **MIME-Type**.

```
final String PALMUSERAGENT      = "Mozilla/3.0 (compatible; HandHTTP 1.1";
final String IEUSERAGENT        = "MSIE 5.";
final String WINCEUSERAGENT      = "Windows CE";
final String WMLWAPUSERAGENT    = "Nokia-WAP-Toolkit/1.2";

final String IEDEVICECHOICE      = "Internet Explorer 5.0";
final String WINCEDEVICECHOICE   = "Hand Held Windows CE";
final String PALMDEVICECHOICE    = "Palm Pilot HandWeb";
final String WMLDEVICECHOICE     = "WML WAP Phone";
```

Figure 133. The key value pair user-agent/device

```

final String JPEGTYPE      = "image/jpg";
final String GIFTYPE       = "image/gif";
final String HTMLTYPE      = "text/html";
final String XMLTYPE       = "text/xml";

```

Figure 134. The MIME type

There are four JavaBeans provided by Transcoding Publisher and used in the Transform Tool, they are shown in Figure 135.

```

ImageTranscoderBean imageTranscoderBean;
HtmlReducerBean htmlReducerBean;
HtmlHandlerBean htmlHandlerBean;
XmlHandlerBean xmlHandlerBean;

```

Figure 135. The Transcoding Publisher JavaBeans in Transform Tool

Each JavaBean relies on property files to provide it with preferences regarding how transcoding should be done for a particular device, what stylesheets are registered, etc.

The Transform Tool gets the device and the MIME type from the user and calls the appropriate bean to transcode it.

For more information about transcoding using the JavaBeans model, see Chapter 8, “Transcoding with JavaBeans” on page 121.

9.2 The Request Viewer

The Request Viewer is a visual tool for monitoring the operation of the transcoding server. You can view which MEGs and MEGlets are registered with the transcoding server, along with the configuration information for the plug-ins.

The Request Viewer is particularly useful as a debugging tool, as it enables you to monitor the flow of requests through the server and observe which plug-ins are triggered and when they are triggered. For each transaction, the Request Viewer also displays the header and content information as they are manipulated by the plug-ins.

Note

The Request Viewer tool is only available when IBM WebSphere Transcoding Publisher is configured to run as a proxy server.

To start the Request Viewer, do the following:

- At the command prompt, change directory to the IBM WebSphere Transcoding Publisher installation directory.
Enter `RunTranscoding -g` at the prompt.
- Windows NT users also have the option of starting the tool by selecting:
Start->Programs->IBM Transcoding Publisher->Toolkit->Request Viewer.

Note

When you start the Request Viewer, you are also starting the transcoding server. If an instance of IBM WebSphere Transcoding Publisher is already running (for example, as a service in the Windows NT environment), the Request Viewer will encounter port conflict errors as it attempts to initialize its own instance of the transcoding server.

These errors will be obvious if you start the Request Viewer from a command prompt, but if you start it from the Start menu, the errors might not be immediately apparent. In this case, the tool will simply not display any request processing activity, even though it might be clear from directing browser requests to the transcoding server that content is being transcoded. To avoid this problem, make sure IBM WebSphere Transcoding Publisher is not running before you start the Request Viewer.

Warning: Due to its resource requirements and potential degradation, it is recommended that the Request Viewer should only be used for debugging purposes. Be careful when working in a production environment since this is a development tool and it should not be used in a production environment.

9.2.1 IBM WebSphere Transcoding Publisher Request Viewer

The IBM WebSphere Transcoding Publisher Request Viewer enables administrators and developers of transcoders, XML stylesheets, and preference profiles to view the operation of transcoders installed in a particular IBM WebSphere Transcoding Publisher server.

The Request Viewer provides two views of the server:

- Server Configuration

A view of the transcoding server's configuration of registered sublayers, plug-ins, and **Monitor-Editor-Generator (MEG)** groups and the transcoding server's tracing information and messages.

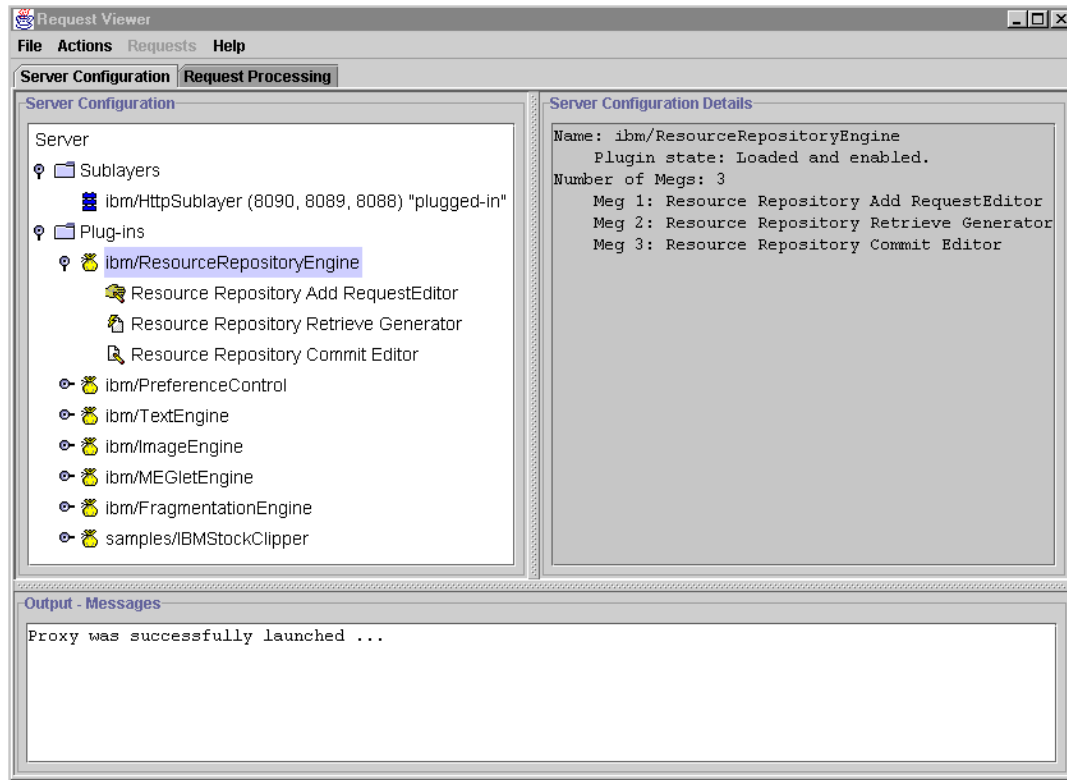


Figure 136. Server Configuration

- Request Processing

A dynamic view of the flow of requests through the transcoding server and the transcoding processes for each request.

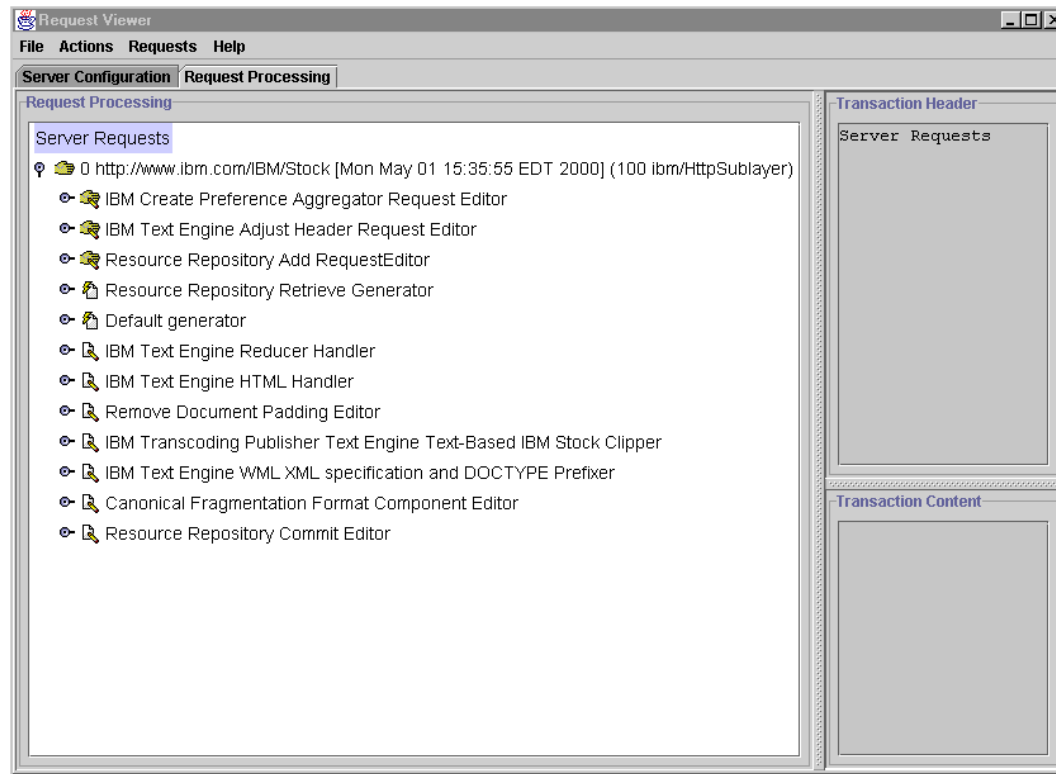


Figure 137. Request Processing

9.2.1.1 Viewing Server Configuration

The Server Configuration view shows the current state of the IBM WebSphere Transcoding Publisher server. The tree view, displayed under Server Configuration, lists the components of the transcoding server. Transcoding server components are grouped as Sublayers and Plug-ins.

- Sublayers

The Sublayers node contains information about the currently loaded sublayers. Sublayers are protocol specific. In most cases, you will see only information about the HTTP sublayer like: ibm/HttpSublayer (8090, 8089, 8088) “plugged-in”.

- Plug-ins

The Plug-ins node contains information for all registered plug-ins and the MEGs contained in a plug-in. Below each plug-in listed in the tree, all of its associated MEGs are represented by individual nodes.

- Request editor
- Generator
- Editor
- Monitor

The details shown for plug-ins are:

- The name of the plug-in
- The state of plug-in (loaded and enabled, etc.)
- Number of MEGs
- List of MEGs

The details shown for MEGs are:

- The name of the MEG
- MEG type
- Condition
- Priority
- State

9.2.1.2 Viewing Request Processing

The Request Processing view shows the flow of requests through the transcoding server and the invocation and the execution of the plug-ins and their associated MEGs as the request is processed.

The tree view, displayed under Request Processing, groups each request and the MEGs that are used to process this request. For each MEG the input and output information is also available. When transaction tracing is enabled the Request Processing view displays information about the individual transactions in a tree in the left part of the view.

As the transaction is being processed, MEGs change the header and the content, and header information displays in the Transaction Header Pane. The data displays in the Transaction Content pane. This data is in either binary or text format depending on the type of transcoding that is being performed.

9.2.1.3 Other functions in Request Viewer

There are a few other functions we need to be aware of when we use Request Viewer. These are invoked using the following pull-down menus:

- File

- Actions
- Requests
- Help

Basically, the functions above are for tracing and debugging, messages and also to clear the message area and request processing views.

One thing we have to remember is that when viewing a request or a trace using Request Viewer is that we can only view the information. So if you are looking for a specific item stop any requests. Otherwise you might override the content of the view.

Note

Since Request Viewer does not support copy and paste from the menu bar, you need to use Ctrl-C and Ctrl-V keys.

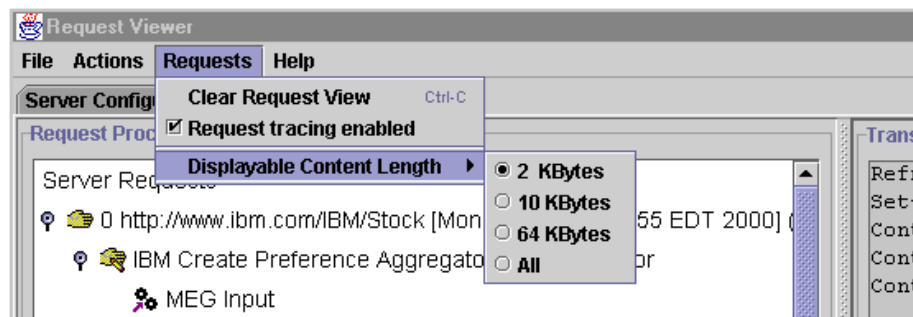


Figure 138. Functions that can be performed for the requests

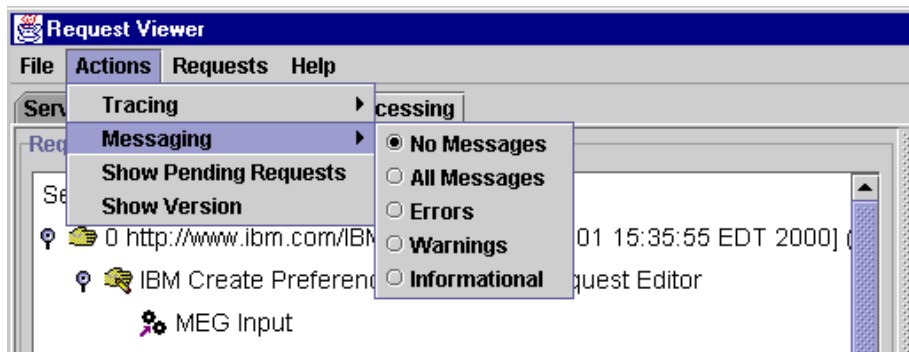


Figure 139. Messaging options

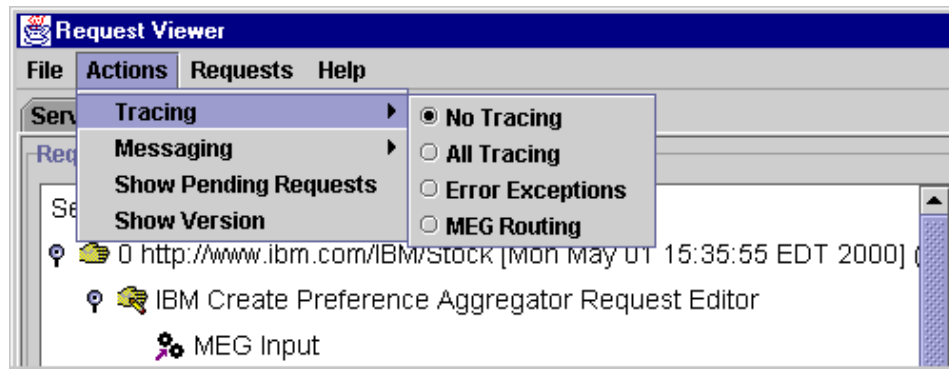


Figure 140. Tracing options

Figure 140 shows what functions we can perform using Request Viewer:

- Tracing
 - No tracing
 - All Tracing
 - Error Exceptions
 - MEG Routing
- Messaging
 - No Messages
 - All Messages
 - Errors
 - Warnings
 - Informational
- Show Pending Requests
- Show Version

The results are shown in the Output Messages pane so you must select the Server Configuration view when you want to see the results.

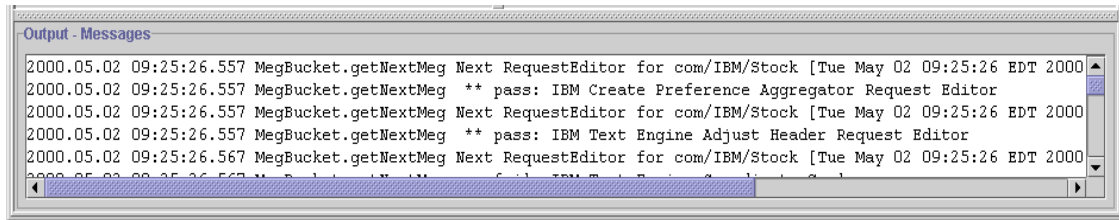


Figure 141. Output Messages pane showing all messages

When you want to see all requests in the Request Processing view you have to enable the Request Tracing by checking the tick box. You can also stop the tracing by unchecking the tick box.

Here is a sample of the Output Messages pane after using a WAP device to request www.ibm.com.

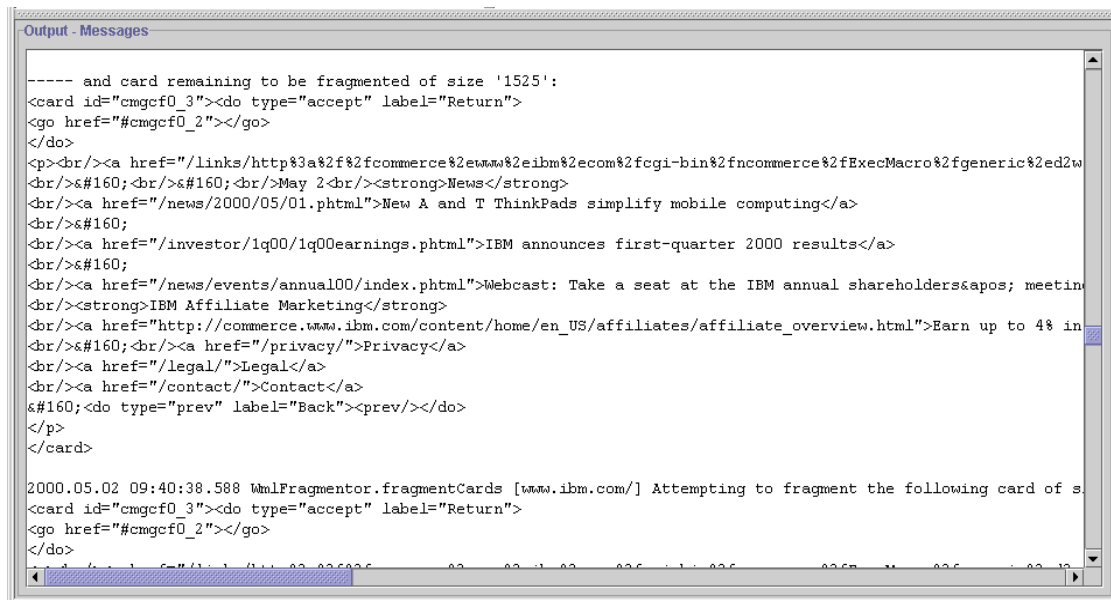


Figure 142. Output messages showing one card for WML transcoding

The Request Viewer can be used as an overall tool to learn how IBM WebSphere Transcoding Publisher works, or you can use it for problem determination and also searching for bottlenecks in the system.

Because you have to stop the service in NT when starting the Request Viewer you have to remember to start the service again when you stop

working with the Request Viewer. Our recommendation is also not to use Request Viewer on a production server, because of the performance degradation.

Also, if you make changes (for example, to a preference profile) through the Administrator's Console, the Refresh Server function will not update the Transcoding Publisher server. If you want changes made in this way to take effect, you must restart the Transcoding Publisher server.

9.3 Creating preference profiles

The Create Profile wizard is designed to help you create a new preference profile for use with Transcoding Publisher. Although a number of preference profiles are supplied with Transcoding Publisher to provide support for most commonly used device and network types, you may find that IBM-supplied profiles do not entirely meet your needs. For example, you might have written a new transcoder intended for a device type that Transcoding Publisher does not currently support, or you might simply wish to create profiles for well-known devices but tailor the default values and configurable aspects of the preferences.

Transcoding Publisher uses preference profiles to represent the characteristics of devices, networks and a default user profile to represent organizational policies. Each profile tells Transcoding Publisher how to treat documents that will be delivered to that device or over that network.

A preference profile can represent a particular type of device, such as the IBM WorkPad, or a particular network type, such as a wireless network.

9.3.1 Using the Create Profile wizard

To start the Create Profile wizard, open the command prompt, change directory to the directory in which Transcoding Publisher is installed and enter `CreateProfile`.

Windows NT users can select **Start->Programs->IBM Transcoding Publisher->Toolkit->Create Profile**.

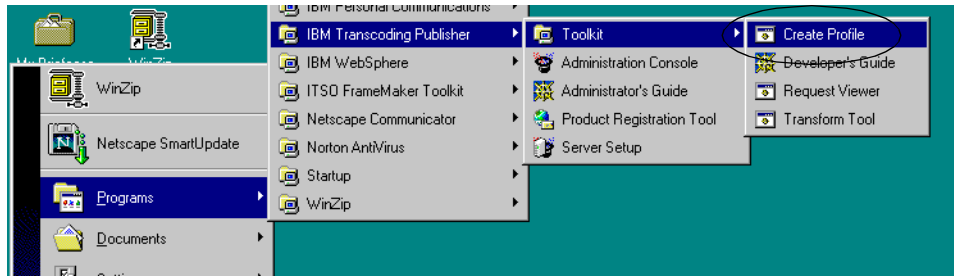


Figure 143. Selecting the Create Profile wizard

Specify the file location, a name and description that will help you or your administrators locate the profile in the Administration Console.

The wizard will display a page showing all the preferences available for that type of profile. You will be able to select the preferences that you want to include and customize the default values accordingly. You can also select if you want that preference to be configurable by the Administration Console with the “*configurable?*” option.

In a preference profile there are several selectable preferences and for each preference, you can specify:

- Whether the preference should be included in this profile. If a preference is not important for this network or device type, do not include it. This will prevent this profile from overriding a value in another profile and this is also a performance issue.
- Whether the preference should be configurable through the Administration Console. For example, you might want to include a preference in the profile, but do not want administrators to be able to change the value you set. Any preferences that you specify as configurable in the wizard are automatically displayed in the Administration Console after the profile has been registered.
- The value for the preference. The default value is provided, but you can change it to any value appropriate for this network or device type. Values can be strings, Boolean or decimal values depending on the preference.

You can specify that a value is configurable, and provide a value for it, only if you have specified that the value is to be included.

After creating the profile, you can distribute it to your IBM WebSphere Transcoding Publisher administrator or make it available to other organizations that are using IBM WebSphere Transcoding Publisher. If you

are creating a profile for use with a custom transcoder, you should include the profile in your transcoder package.

9.3.2 Creating a network preference profile

To create a network preference profile open the Create Profile tool as illustrated in Figure 143 on page 175.

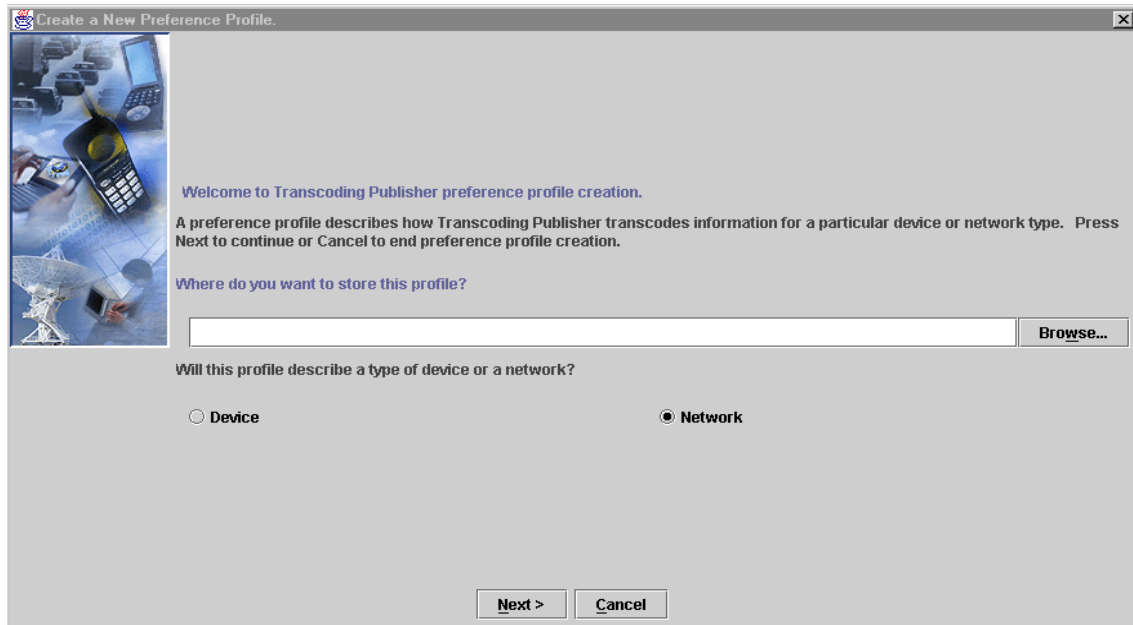


Figure 144. Creating a new network preference profile

The invocation of the tool is a batch file called CreateProfile.bat under the *IBMtrans* install directory.

The CreateProfile.bat file looks like this:

```
@echo off
setlocal
set DEBUG=false
set DIRECTORY=.
set
CLASSPATH="%DIRECTORY%;%DIRECTORY%\lib\ras.jar;%DIRECTORY%\lib\swingall.jar;%DIRECTORY%\lib\js.jar;.;%TP_JAVA_HOME%\lib\classes.zip"
set PATH=%TP_JAVA_HOME%\bin;%DIRECTORY%;%DIRECTORY%\bin;%PATH%;
:rungui
if "%1"=="-d" goto runWithDebug
jre -cp %CLASSPATH% com.ibm.transform.gui.PreferenceCreatorWizard
```

```
goto finish
:runWithDebug
jre -cp %CLASSPATH% com.ibm.transform.gui.PreferenceCreatorWizard -debug
:finish
endlocal
```

The batch file includes a debugging capability. Because we use a Java run-time environment we will have two windows opened, one of which is the command prompt window and the other the actual wizard window. You should not close the command prompt window when creating profiles or the other session will be closed and you will lose all your definitions.

The next step is to define where to install the profile and what to call it. Click **Browse**. The default directory is *IBMTrans*.

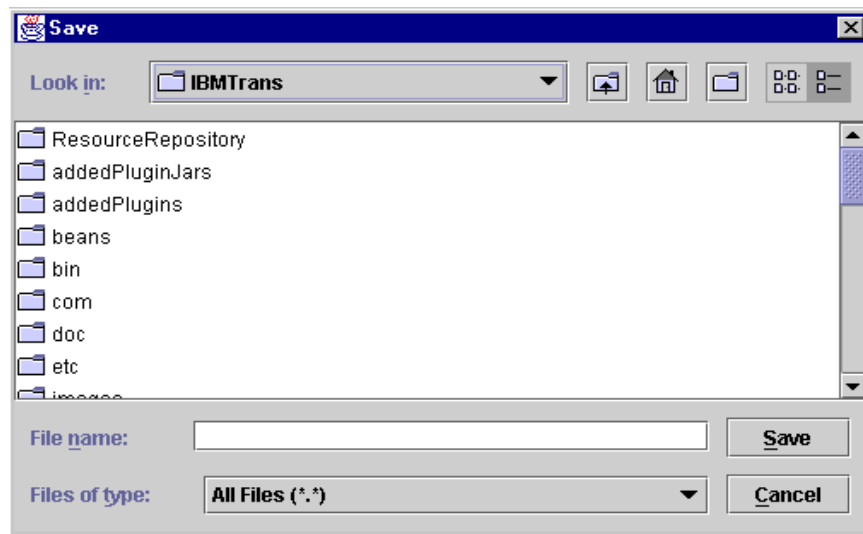


Figure 145. Browsing the default directory *IBMTrans*

Select *etc* directory.

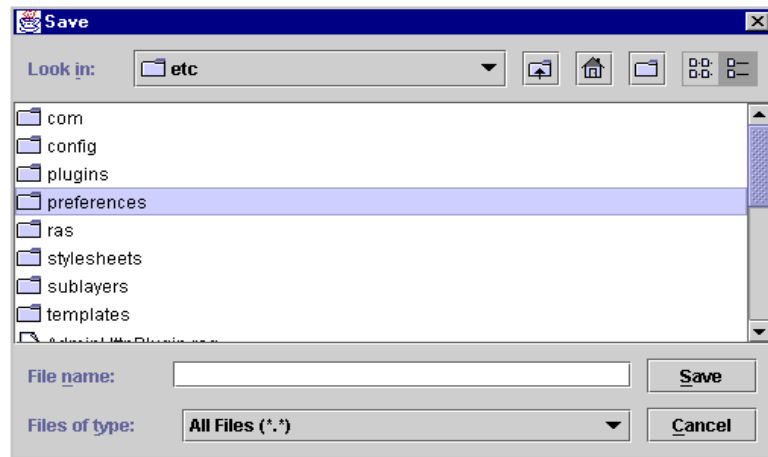


Figure 146. IBMTrans\etc subdirectory

Select the **preferences** directory.

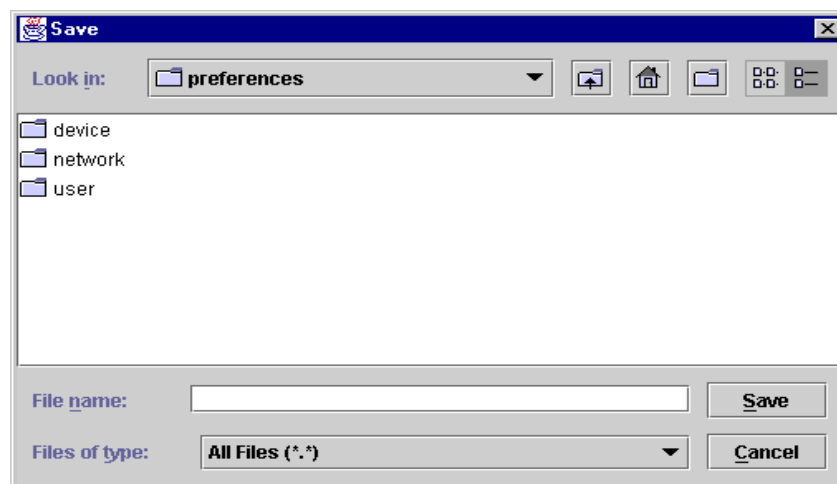


Figure 147. IBMTrans\etc\preferences subdirectory

Select the **network** directory.

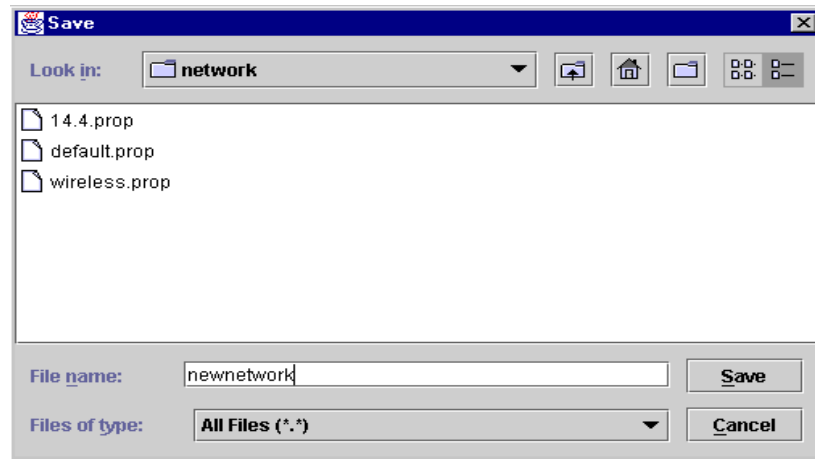


Figure 148. IBMTrans\etc\preferences\network subdirectory

We will call this profile `newnetwork`. An alternative way is to type the full path including the filename for example:

`C:\Program Files\IBMTrans\etc\preferences\network\newnetwork.prop`

Then click **Save**.

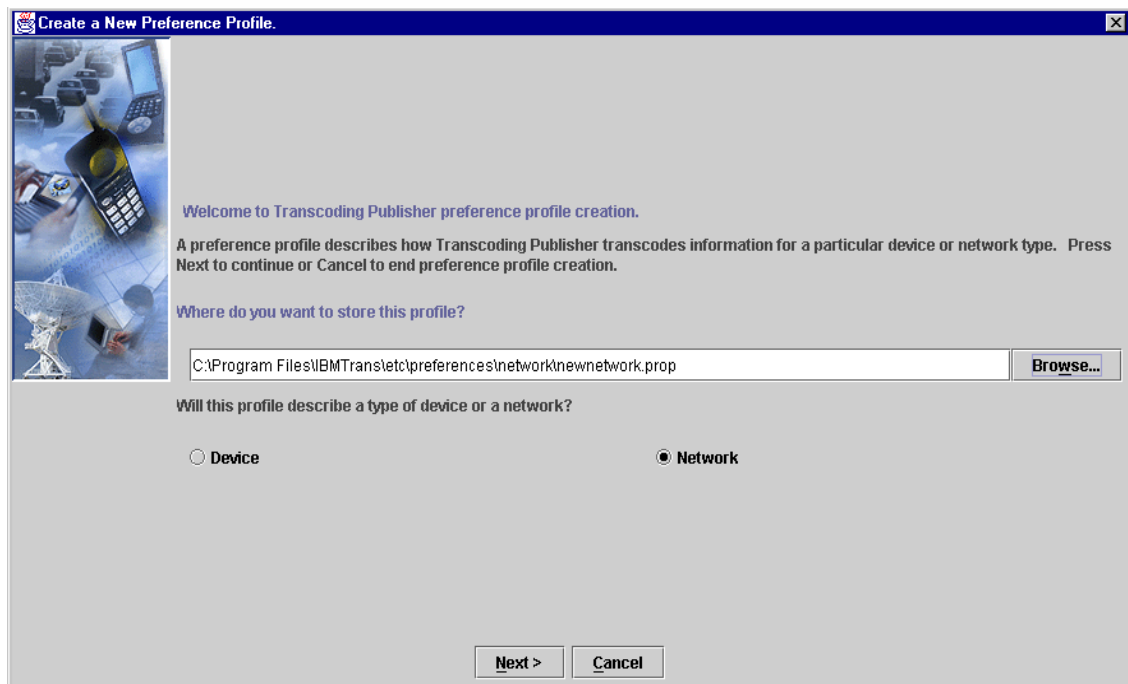
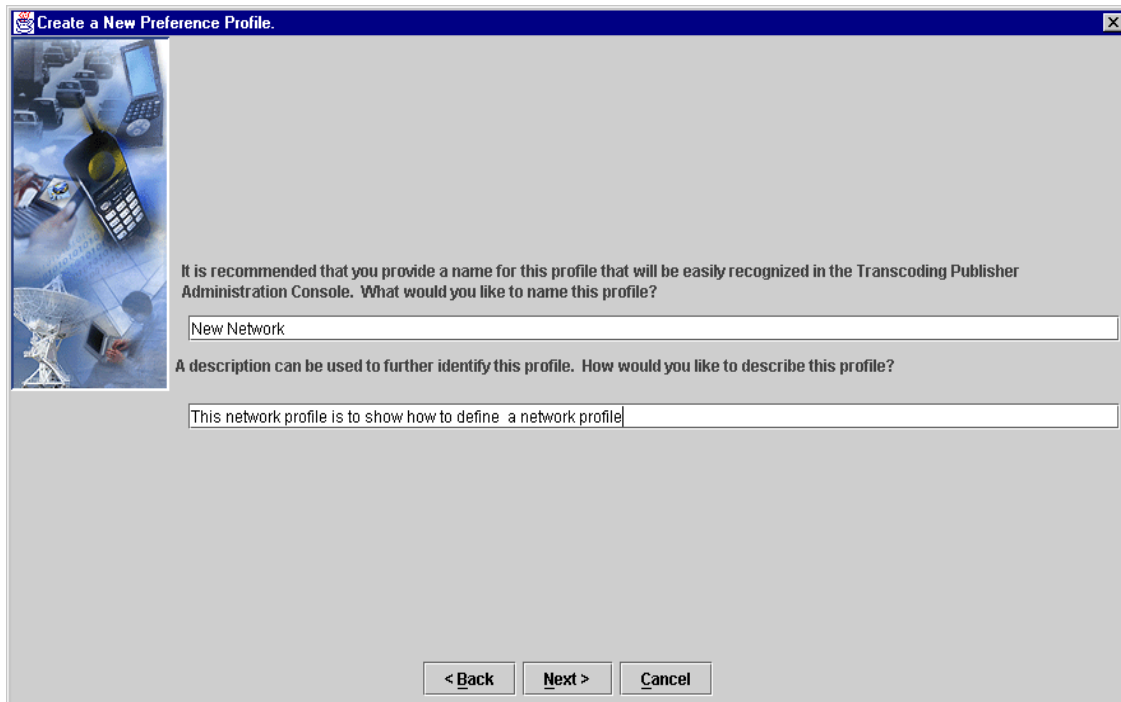


Figure 149. The name for the new network profile

Click **Next**.



Create a New Preference Profile.

It is recommended that you provide a name for this profile that will be easily recognized in the Transcoding Publisher Administration Console. What would you like to name this profile?

New Network

A description can be used to further identify this profile. How would you like to describe this profile?

This network profile is to show how to define a network profile

< Back Next > Cancel

Figure 150. Naming the profile

In the window above, we have to provide a name for the profile. The upper field is the name you see after you have registered this profile and the second field is for more detailed information about this profile. We enter *New Network* for the name and select network profile.

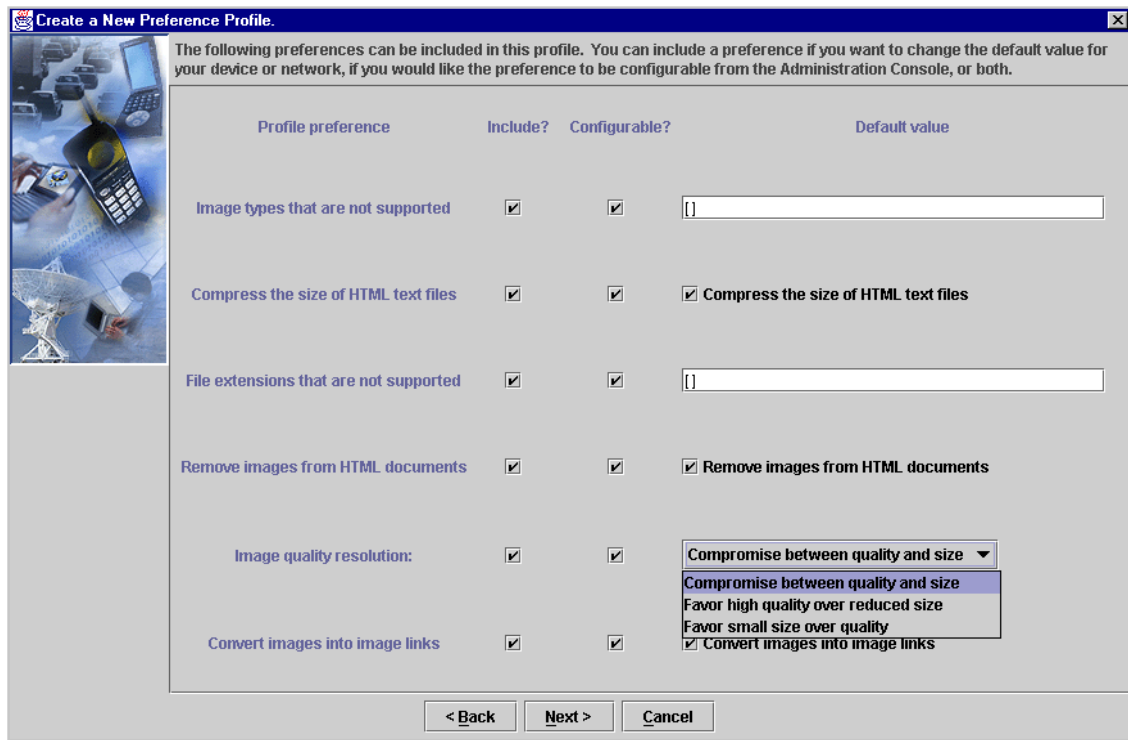


Figure 151. Creating a new network preference profile

We have the option to select the following preferences for that profile:

- Image types that are not supported

Here we enter those image types we do not want to be sent over the network, such as bmp, tga, and pct. All entries must be separated by comma.

- Compress the size of HTML files
- File extensions that are not supported

Here we enter those file extensions we do not want to be sent over the network, such as wav, mpg, etc. All entries must be separated by comma.

- Remove images from HTML documents
- Image quality resolution
 - Compromise between quality and size
 - Favor high quality over reduced size

- Favor small size over quality
- Convert images into image links

We have the option to select which of these preferences will be shown in the Administration Console as configurable. It might be suitable to leave most of these as configurable for testing purposes. Sometimes you might want to modify these on the fly because of extra charges for the network changes or other similar reasons.

When you have made your definitions click **Next**.

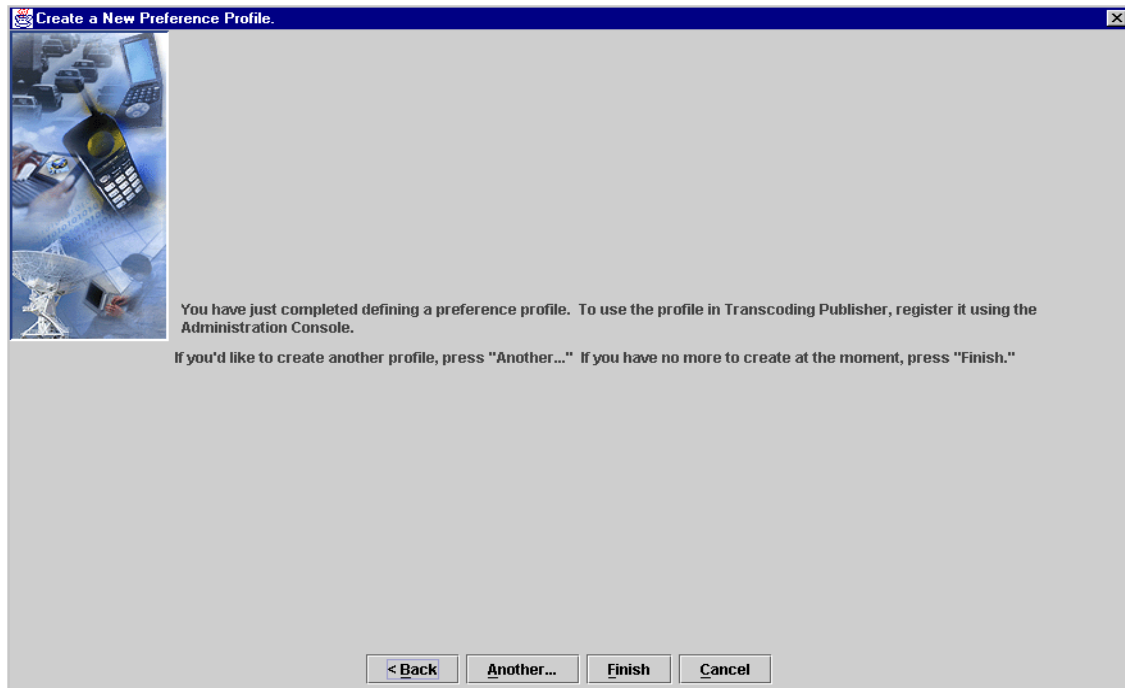


Figure 152. Finish creating network profile

When you are finished click **Finish**.

Note

If you select Another, you will not create another profile with the same definitions. You will be editing the one that you have just created. The solution for this problem is to finish and start the Create Profile utility to create the second network profile.

When you finish the process of creating a new profile, you have to register this new network profile for the IBM WebSphere Transcoding Publisher. For details on how to register your new network profile see 12.4.1, “Registering network preference profile” on page 269.

9.3.3 Creating a device preference profile

To create a device preference profile, open the Create Profile tool:

Programs --> IBM Transcoding Publisher --> Toolkit --> Create Profile.

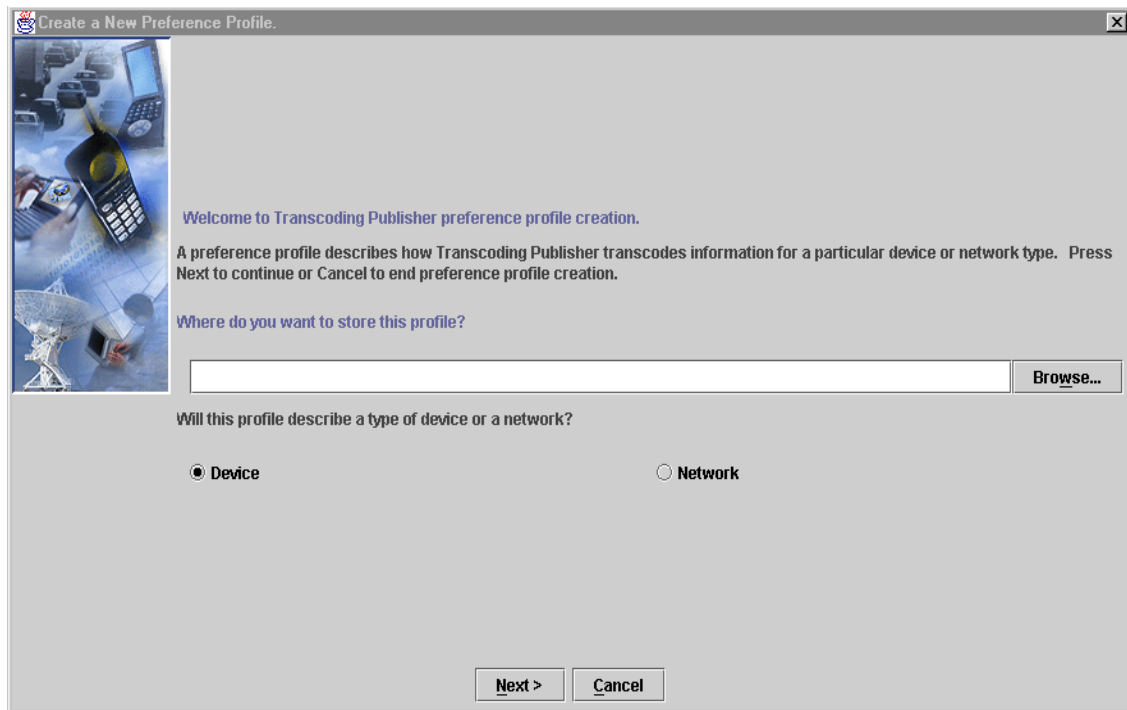


Figure 153. Create a device preference profile

Next we have to define where we want to save the device profile and provide a name for it.

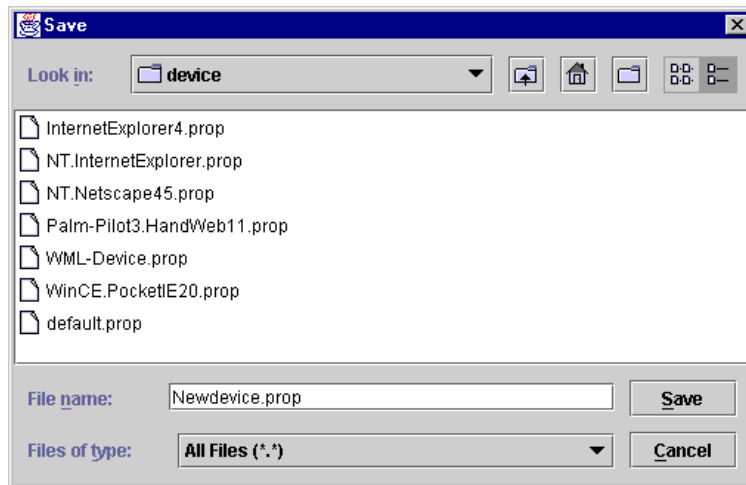


Figure 154. IBMTrans\etc\preferences\device subdirectory

The default directory is IBMTrans. Select the icon for directory *etc* then select the icon for directory *preferences* and then select the icon for *device*. We use Newdevice.prop for the file name then click **Save**.

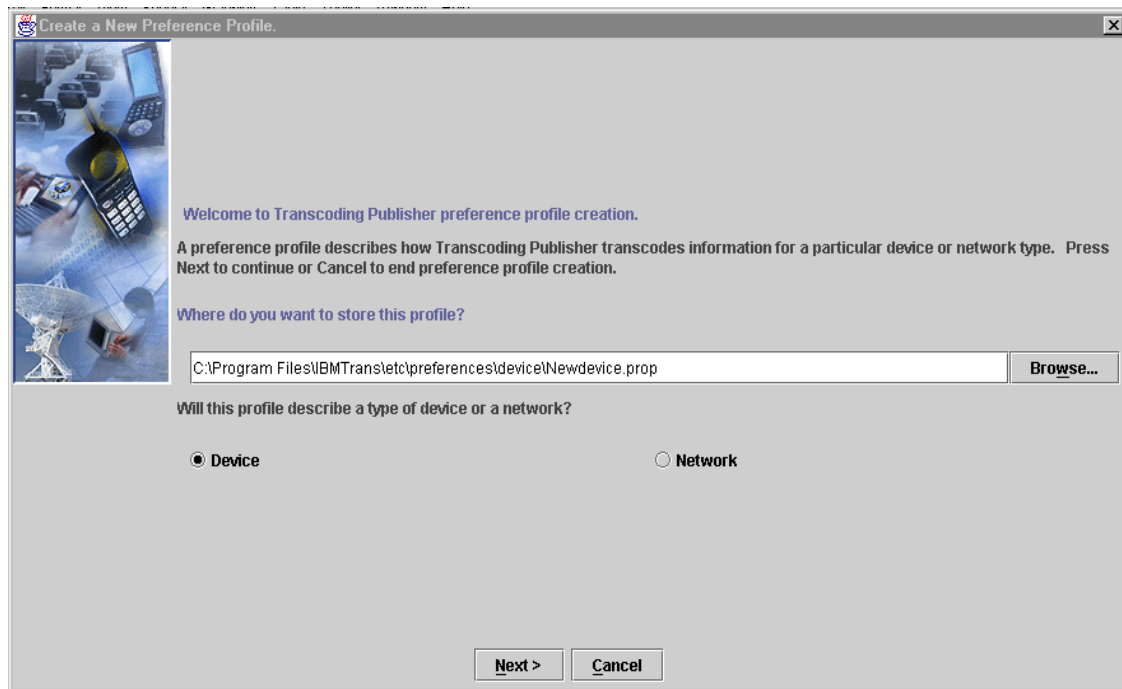


Figure 155. Full path and filename for the new device profile

Another way is to enter the full path including the filename for example:

C:\Program Files\Files\IBMTrans\etc\preferences\device\Newdevice.prop.

Create a New Preference Profile.

It is recommended that you provide a name for this profile that will be easily recognized in the Transcoding Publisher Administration Console. What would you like to name this profile?

New Device

A description can be used to further identify this profile. How would you like to describe this profile?

New Device profile to show how to create new device profile

< Back Next > Cancel

Figure 156. Naming the device profile

Now we have to name the device profile. The upper field will be the name you will see after you have registered this profile in the Administration Console. The second field is for more detailed information for the profile.

Enter *New Device* for the name and *New Device profile to show how to create new device profile* for the more detailed information field.

Next, we have to define a value for the user-agent. This is shown in Figure 157.

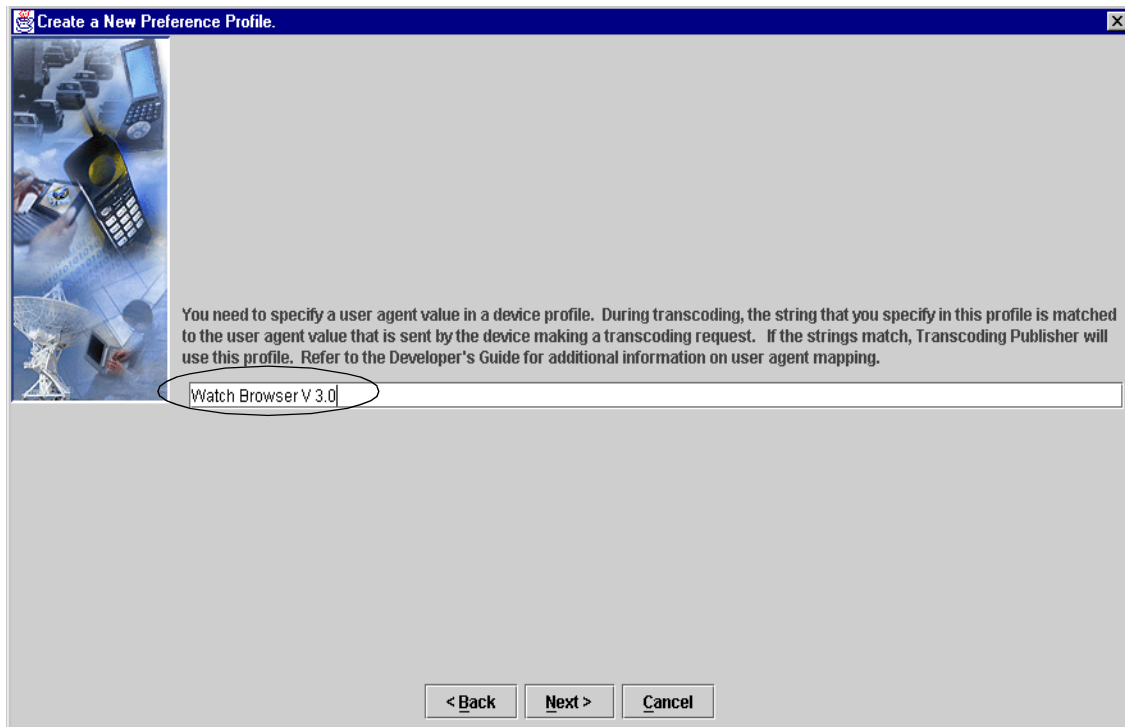


Figure 157. User-agent field value

This value is the string value or part of the string value that is included in the request header from the new device that we are creating the profile. This value must be unique.

Whenever IBM WebSphere Transcoding Publisher gets a request it will first see if there is a device for which the user-agent field matches the value in the request. It should be delivered from the manufacturer with the device.

If you do not have this value, there are two ways to determine it.

1. You can use the Snoop transcoder to discover the correct value. This tool will report back the request information that was sent by the client along with any request attributes that have been set prior to this point (including aggregated preference set up by the transcoding service).
2. Use the Request Viewer tool to find out the user-agent information. We show here the result when using Microsoft Internet Explorer Version 2.0:

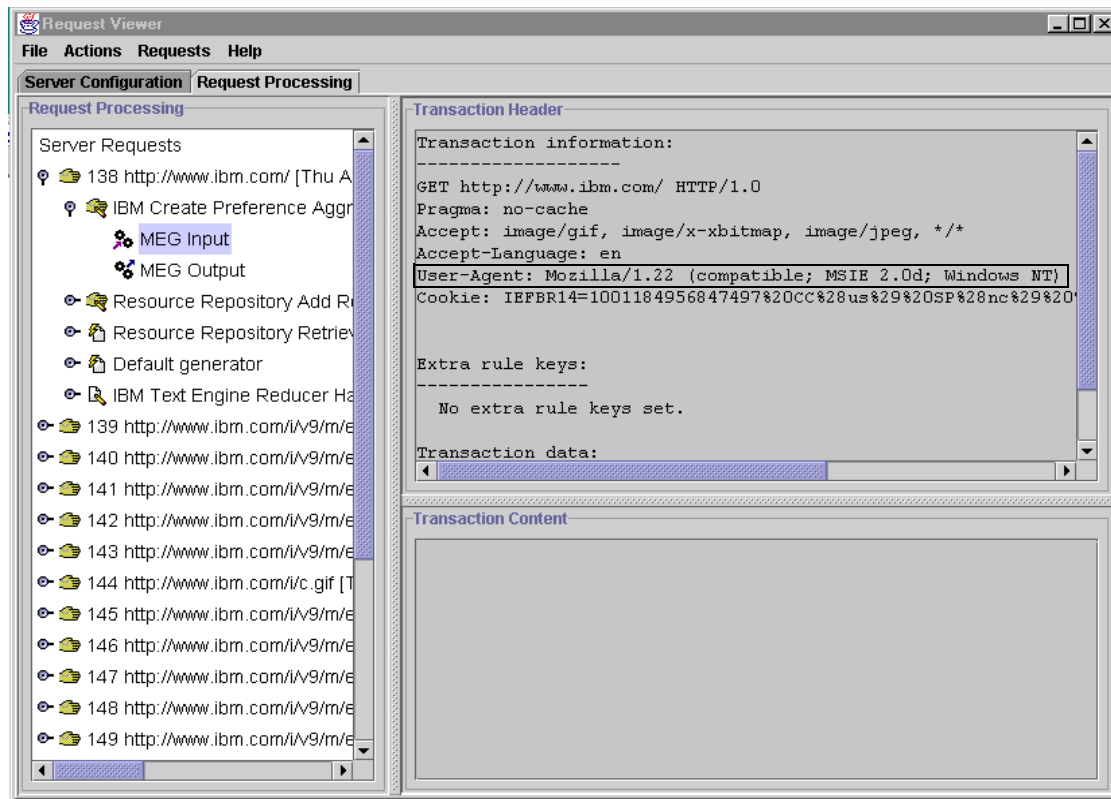


Figure 158. Request Viewer used to find out value for the user-agent field

In this example we enter the following value: *Watch Browser V3.0*. In the following page we define the transcoding options we want to use with this device.

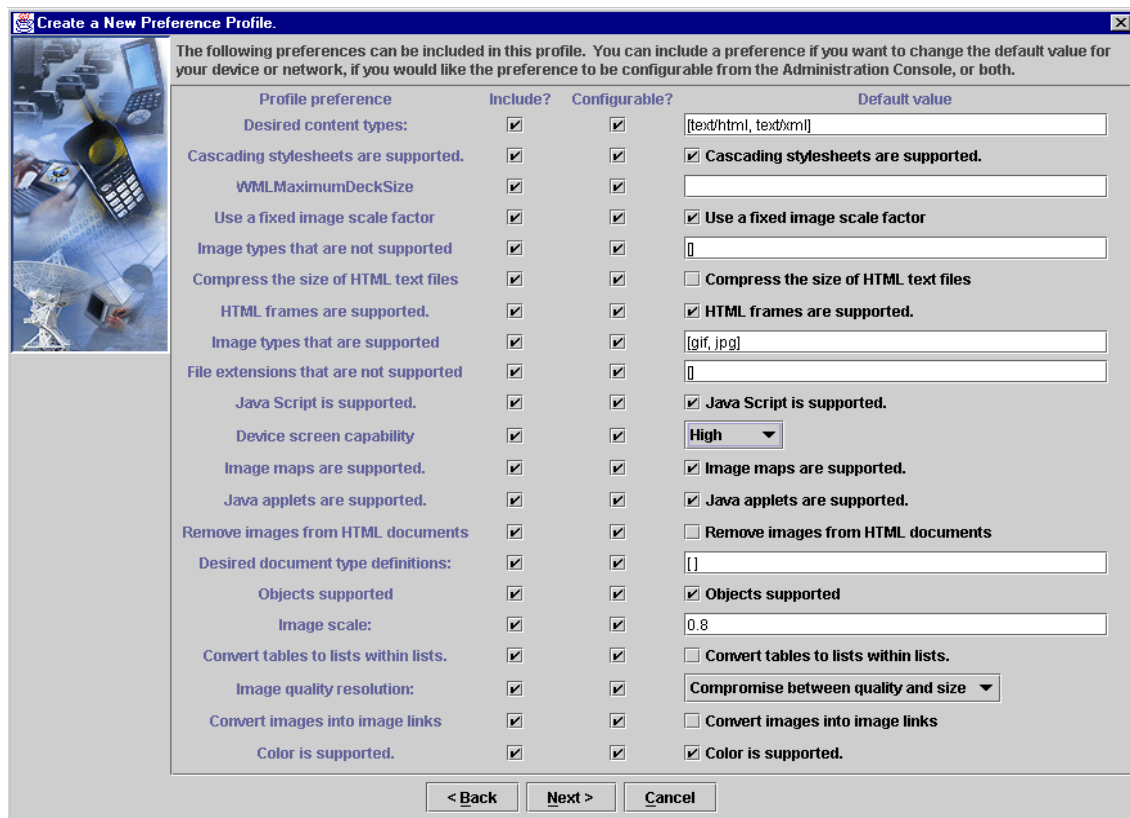


Figure 159. Defining device preference profile options

The following options are available:

- Desired content types
- Cascading stylesheets are supported
- WML maximum deck size
- Use fixed image scale factor
- Image types that are not supported
- Compress the size of HTML text files
- HTML frames are supported
- Image types that are supported
- File extensions that are not supported
- Java Script is supported

- Device screen capability
 - High
 - Medium
 - Low
- Image maps are supported
- Java applets are supported
- Remove images from HTML files
- Desired document type definitions
- Objects supported
- Image scale
- Convert tables to lists within lists
- Image quality resolution
 - Compromise between quality and size
 - Favor high quality over reduced size
 - Favor small size over quality
- Convert images to image links
- Color is supported

In our example we wanted to select all options because we wanted to make everything configurable by the Administration Console.

When you have finished selecting the preferences that you will use, click the **Next** button.

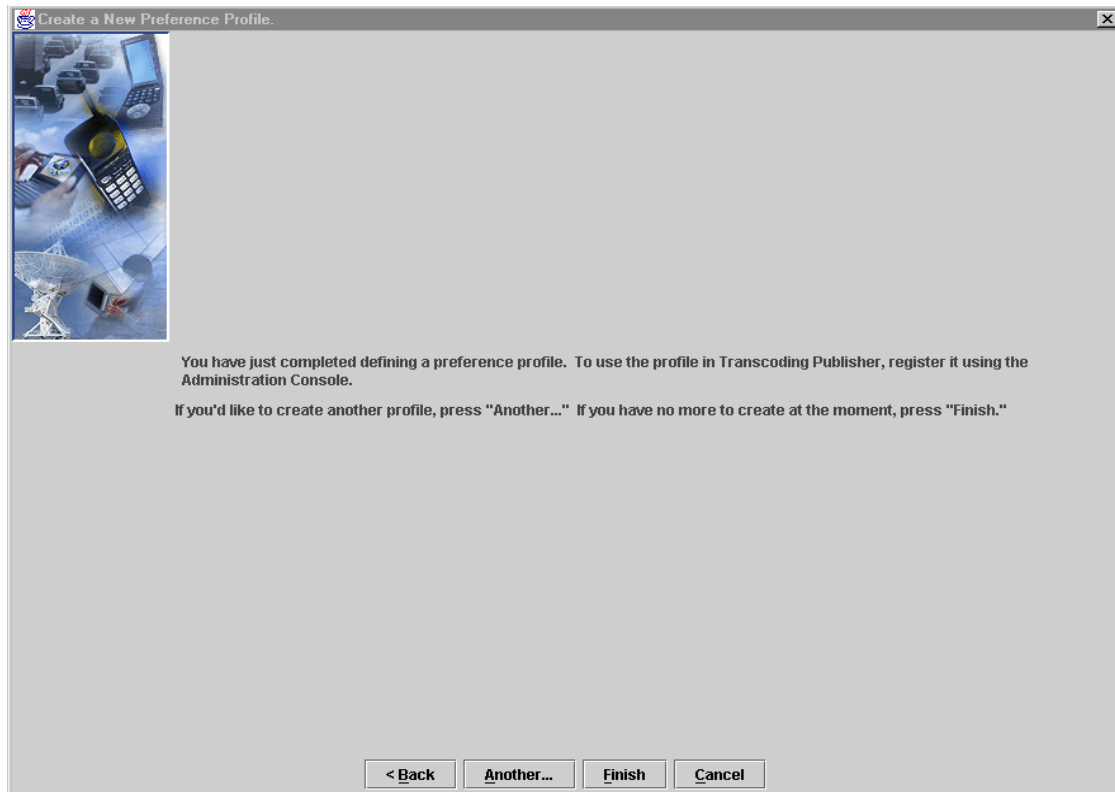


Figure 160. Finish creating device profile

We have now finished creating the device profile.

Click **Finish**.

The next step is to register the profile for IBM WebSphere Transcoding Publisher. For information on how to register your new device profile see 12.4.2, “Registering a device preference profile” on page 276.

9.4 The Snoop tool

The Snoop tool is a sample “MEGlet” and it actually provides function similar to a servlet already available in WAS 2.0.3 with name SnoopServlet (at <http://<server-name>/servlet/SnoopServlet>). The snoop tool however can be run in the WTP framework independent of WAS. The tool also runs in WAS, but since WAS 2.03 supports an older version of the servlet API, this is probably not recommended.

One of the values we have to define for the new profile is the user-agent. It is a string value or part of the string value that is included in the request header from the new device and must be unique. For additional information about this field see Chapter 12, “Administration” on page 261 and Chapter 8, “Transcoding with JavaBeans” on page 121.

In WTP, there are basically two different ways to determine the value of the user-agent field. You can use the Request Viewer tool (see 9.2, “The Request Viewer” on page 166) or you can use the Snoop tool. In this section, we describe how you can use the Snoop tool for this purpose.

Important: You should also be aware that in some cases, one device can have multiple user-agents associated with it. This is unlikely to be the case with a phone for example where software choices are limited, but a Palm Pilot can have multiple browsers running at the same time. Just like a desktop PC can have both Netscape and Internet Explorer, a Palm Pilot can have both HandHTTP and PalmScape.

9.4.1 Using the Snoop tool

The Snoop tool provides information about which user-agent will be actually used by WTP. It reports back the request information that was sent by the client in the request. The Snoop tool is distributed in the `toolkit\meglets\Snoop` directory.

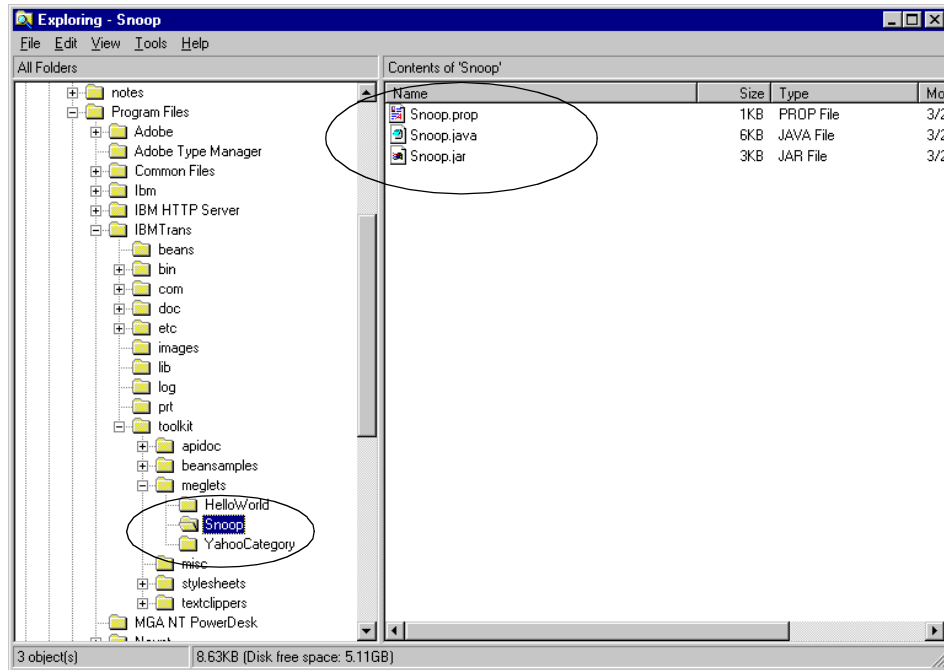


Figure 161. The Snoop tool path

The tool does not require any input parameters and it produces an HTML report listing the request information, request headers (including the user-agent field) and the init parameters.

Note

The Snoop tool is a MEGlet of type generator and it runs in the WTP proxy model. WTP requires configuration information for the MEGlet (for details see 9.5.2.3, “Specifying MEGlet configuration information” on page 200).

However, if you want to run the snoop tool as a servlet you may want to execute the following procedure:

1. If required, compile the Snoop.java tool. Keep in mind that when running this tool as a servlet, it requires the servlet.jar in the classpath to compile.
2. Make the servlet visible to WAS. For example, copy the Snoop.class file to the WebSphere directory <install_path>\AppServer\servlets\.
3. From a Web browser, invoke the tool. For example, use the following URL:

`http://<server-name>/servlet/Snoop`

Figure 162 shows the display results for a desktop Netscape browser. The user-agent field for this client can be found in the request header (Mozilla/4.61 in this sample display).

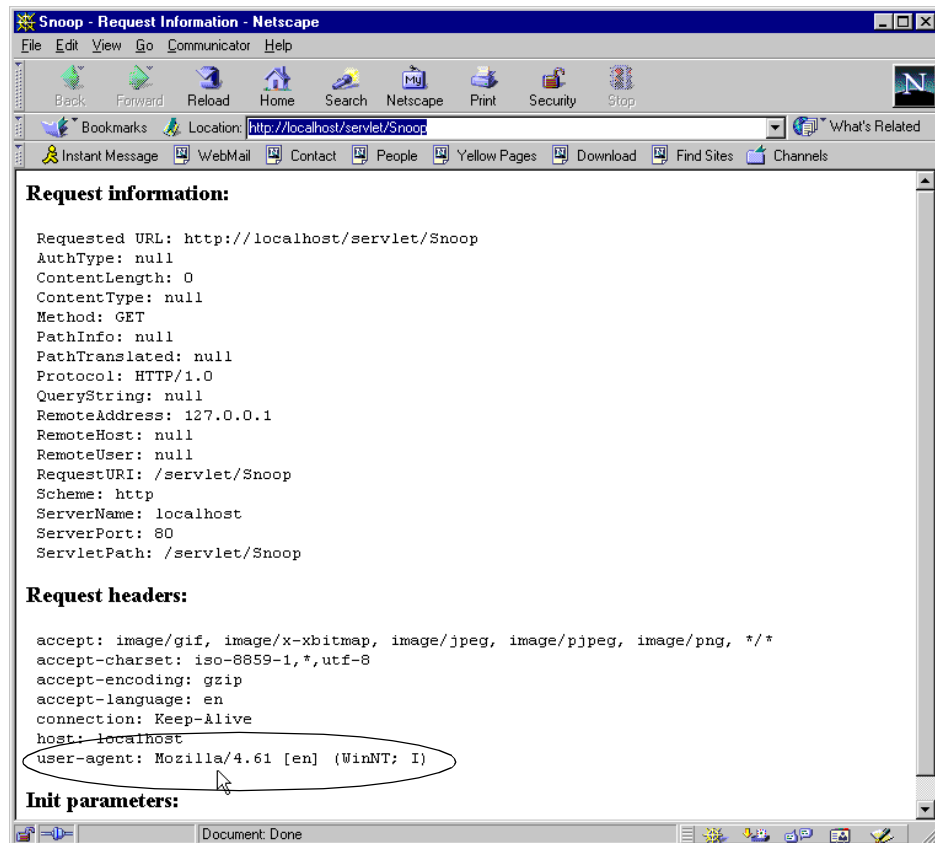


Figure 162. Netscape browser - Snoop tool results

Figure 163 on page 196 shows the display results for a desktop Internet Explorer browser. The user-agent field for this client can be found in the request header (Mozilla/1.22 in this display).

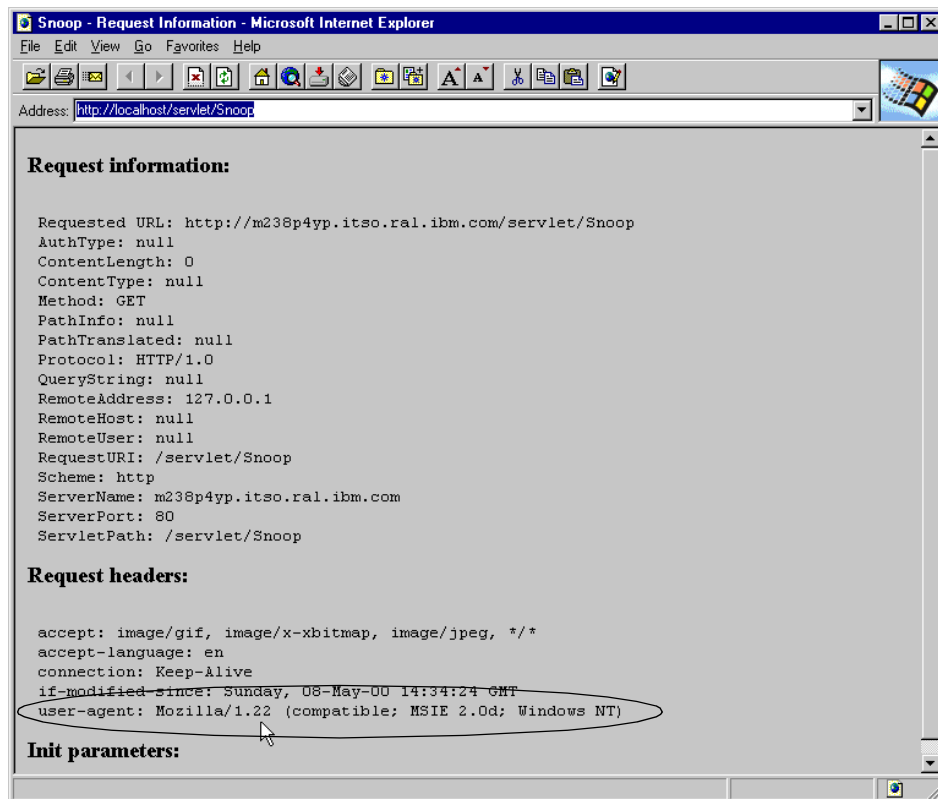


Figure 163. Internet Explorer browser - Snoop tool results

9.4.2 Transcoding Snoop content

In some cases you will need to run this tool from other clients such as WAP phones, Windows CE and Palm devices. Since this tool produces HTML content, you may need to transcode the output. For example, a WAP phone will require the HTML code to be transcoded into WML and so on.

There are several ways you can do this for example:

1. Run WTP as a proxy server. See Chapter 5, "Running as a stand-alone network proxy" on page 63 for details.
2. Run WTP as WAS filters. In this scenario, you will need to enable the Snoop servlet for transcoding and define WTP to run as a WAS filter for MIME-type text/html. See Chapter 7, "Transcoding with WAS filters" on page 93 for details.

3. Upgrade the Snoop tool to invoke WTP JavaBeans. See Chapter 8, “Transcoding with JavaBeans” on page 121 for details on how you can invoke WTP JavaBeans from this servlet.

As a final example, we show how this tool is invoked from a WAP device where HTML-to-WML transcoding is required. From the initial WML page request we can request the execution of the tool running in the WebSphere Application Server environment. The servlet produces the HTML code and WTP transcodes the content to WML which is sent to the WAP device as illustrated in Figure 164. Notice that the user-agent in this case is displayed as Nokia-WAP-Toolkit/1.2.

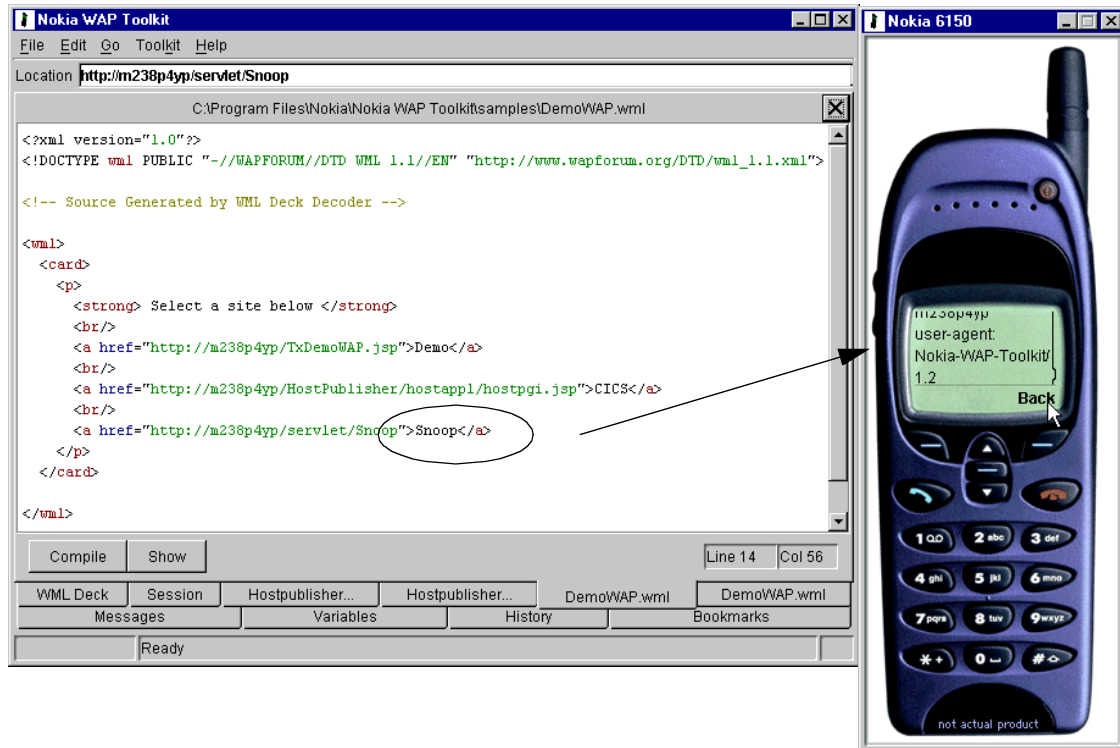


Figure 164. The DemoWAP.wml

9.5 Creating new transcoders for Transcoding Publisher

Transcoding Publisher's framework is built around a pluggable architecture, so the capability of the transcoding server can be easily extended to support new data formats or to meet special transcoding needs. Transcoding

Publisher supports transcoders written either to the *Web Intermediaries (WBI) Version 4.4, API*

(<http://www.almaden.ibm.com/cs/wbi/doc/api/index.html>), or to the *Java Servlet Version 2.1 API*

(<http://java.sun.com/products/servlet/2.1/index.html>). Referred to as MEGs and MEGlets, respectively, these transcoders can be used to edit requests flowing through the framework, to generate a response, to edit the response as it flows back to the client, and to monitor the response. Deciding which API to use depends on the requirements of your transcoder and the environment in which it is intended to run.

Because the Servlet API is well understood and supported in a number of environments, (such as WebSphere Application Server, Apache, and Java Web Server), transcoders written to this API can be easily ported to other environments. Similarly, transcoders written for these other environments can be brought into Transcoding Publisher with few modifications, and they will be able to take advantage of other IBM-supplied transcoders and services, like preference aggregation.

Whereas servlet-based transcoders provide for portability across environments, transcoders using the WBI API can leverage WBI's ability to provide greater control over request processing and access to information in the request itself. In addition, WBI-based transcoders can use other WBI facilities, such as JavaBeans providing various convenient functions.

If you think your development needs are such that the WBI API is a more appropriate approach for your transcoder, you can refer to the WBI Development Kit (<http://www.almaden.ibm.com/cs/wbi/doc/api/index.html>) for detailed information on the API and how it can be implemented.

However, if the portability or environmental advantages of the Servlet API are suited to your transcoder, this section describes the steps required to build and deploy a MEGlet in Transcoding Publisher. Because the concepts of WBI and its MEG-based transcoders are important to the development of MEGlets, it is recommended that you also read WBI Basics (in the Developer's Guide, under Architecture and Components -> Transcoding Framework), if you are not familiar with WBI's concepts.

Note: Running MEGlets in the servlet model is not supported in all versions of WTP. In the servlet model, MEGlets that are written to be compatible with both the 2.0 version of the servlet API (WAS) and the 2.1 version (WTP) can be run as either servlets in WAS or MEGlets in the proxy, without code changes (or even a recompile). For example, the sample TranscodingSamples is an example of such a servlet.

9.5.1 Servlet resources

The Java Servlet API is both widely supported and widely documented. A good starting point for information about servlets is Sun's Web site (<http://java.sun.com/products/servlet/index.html>). In particular, the servlet technical resources page (<http://java.sun.com/products/servlet/technical.html>) provides articles and tutorials on a number of topics, as well as a list of third-party resources such as books and other Web sites. The Java technology zone (<http://www.ibm.com/developer/java>) of IBM's developerWorks Web site also has a variety of resources, downloads, and news items about Java and servlets.

9.5.2 Adding a transcoder

Adding a new transcoder requires the following steps:

1. Write a MEGlet and compile it against the Servlet 2.1 library.
2. Specify the MEGlet configuration information in a property file (for details see 9.5.2.3, "Specifying MEGlet configuration information" on page 200).
3. Package the compiled class files and property file in a Java Archive (JAR) file.
4. If your transcoder requires its own preference profile, create the profile.
5. Register the transcoder and the preference profile (if necessary) with Transcoding Publisher.

9.5.2.1 Writing and compiling the MEGlet

As indicated above, MEGlets can perform any number of functions, singly and in conjunction with other MEGlets through chaining. You can create the MEGlet to act as a request editor, monitor, response editor, or generator.

For examples of how MEGlets can be implemented as generators, editors, and monitors, Transcoding Publisher includes several samples that you can experiment with and modify for your own purposes.

The Servlet 2.1 library is provided with Transcoding Publisher as `<install_path>/lib/servlet.jar`, where `<install_path>` indicates the directory where you installed Transcoding Publisher. Alternately, you can obtain the library directly from Sun. To ensure that your code compiles properly, be sure to add this library to your classpath. For example, on Windows NT you might specify:

```
javac -classpath "%classpath%;C:\Program Files\IBMTrans\lib\servlet.jar"  
MyMeglet.java
```

9.5.2.2 Naming your MEGlet

Because MEGlets used with Transcoding Publisher must have unique class names, IBM recommends conforming to Java's name space approach to help ensure that naming conflicts between MEGlets do not occur. Transcoding Publisher organizes MEGlet class files and property files relative to the `addedPlugins` and `etc/plugins` directories, respectively. If hierarchical names are not employed, the possibility of naming conflicts increases as more transcoders are added.

By following a name space approach with your MEGlet, you minimize both the chance of naming conflicts with other transcoders and the possibility that your transcoder might be inadvertently replaced during registration. For example, Transcoding Publisher places all the class files for new transcoders under the `addedPlugins` directory. If your class is named `com.myCompany.myPackage.myMeglet.class`, the directory structure under `addedPlugins` will reflect the package name and help ensure that your class name does not conflict with others.

For details on how to package your MEGlet files for proper registration with Transcoding Publisher, refer to 9.5.2.5, "Packaging the transcoder files" on page 203.

9.5.2.3 Specifying MEGlet configuration information

Once you have written the MEGlet, there is still configuration information required by Transcoding Publisher to indicate such things as what type of MEG the MEGlet represents or the circumstances under which the MEGlet will be triggered. This information is defined in a property file. The contents of the `Snoop.prop` file, available as part of the Snoop sample, illustrates what can be included in a property file:

```
Class = Snoop
Name = /servlet/Snoop
MEGType = generator
Condition = path = */servlet/Snoop
Priority = 10
InitParameter = [logging = true, level =5]
Description = Sample generator servlet
```

Figure 165. Snoop sample

A complete list of allowed properties and their values is described below.

Table 3. MEGlet properties and description

Property	Description
Class	The name of the class that implements the MEGlet, including the package name, if you specify one for example: Class = com.ibm.transform.toolkit.TransformTool
Name	The name that the MEGlet will be known by in WTP. This is used by the RequestDispatcher class and by the Administration Console. It must be unique for example: /servlet/myServlet
MEGType	The role this MEGlet plays when involved in transcoding transactions. The possible values are: generator, editor, requestEditor and monitor
Condition	A Boolean condition that, when true, causes this MEGlet to be triggered. For example, with the Snoop MEGlet defined by Snoop.prop above, the MEGlet will be invoked if the path in the HTTP request ends with the string "/servlet/Snoop".
MonitorPosition	For MEGlets acting as monitors, this property specifies the location of the monitor in the response editor chain. Possible values are: START - the MEGlet runs before the response is edited (right after the generator runs, producing the response) END - the MEGlet runs after all editing is complete EDITOR name - the MEGlet runs after a particular editor servlet, with name identifying the editor servlet
Priority	The ordering priority relative to other MEGlets. The allowable range for priority is between 100 and 1, with 100 indicating the highest priority and 1 indicating the lowest. If the triggering conditions of more than one MEGlet match the request, the MEGlet with the highest priority is invoked first. The sequencing for multiple MEGlets is the same as that for MEGs, and both MEGlets and MEGs can interoperate on the same request.
InitParameter	Initialization parameters to be supplied to the MEGlet. The format for the parameters is [name = value, ...], with <name, value> pairs separated by commas. Use successive commas (,,) to embed a comma in a parameter. These parameters are specific to your code and can be whatever is meaningful to your MEGlet.
Description	Descriptive text that identifies the MEGlet. This information is displayed in the Administration Console after the transcoder has been registered. You can use whatever description you wish.

9.5.2.4 Setting triggering conditions

A MEGlet is selected to handle a request based on its condition, as defined by the Condition property in its configuration (.prop) file. At most, one generator will produce content in response to a request. Generators whose condition matches the request will be invoked in priority order until one produces a response. A generator can throw a ServletException to indicate that it does not wish to handle the request. If no generators produce a response, a default generator forwards the request on to the server specified in the request.

Request and response editor MEGlets are also invoked based on their condition, in priority order, with each one acting in the manner of a chained servlet to filter the headers and data in the request or response. Monitors are associated with the response chain, and can be run (again based on condition) at various points in the chain. The output stream is not available to Monitor MEGlets.

Conditions are Boolean expressions that test the HTTP request or response headers. The following keywords are defined:

- **method:** The request method for example, GET.
- **protocol:** The request protocol for example, HTTP.
- **query:** The request query for example, v=9&q=code+fixes&x=9&y=12.
- **user-agent:** The user-agent field of the request for example, Mozilla/1.1-(compatible;-MSPIE-2.0;-Windows-CE).
- **host:** The hostname of the server for example, www.ibm.com.
- **path:** The path in the URL for example, /network/mobile/index.html.
- **url:** The fully qualified URL for example, www.ibm.com/network/mobile/index.html
- **port:** The local port for example, 8080.
- **content-type:** The content type of the response for example, text/html.
- **responseCode:** The response code. For example, 200 or 404.
- **server:** The server software that generates the response for example, Apache/1.3.4.

Keywords are followed either by an equals sign (=), which indicates that the argument is case-sensitive, or by a tilde (~), which indicates that the case is ignored. After the equals sign or tilde is a single argument. The argument can be grouped into clauses using parentheses to make a Boolean expression. To negate an argument, precede it with an exclamation point (!). The operators

AND and OR (& and |, respectively) can be used between clauses.

Arguments can include:

- Asterisks (*) used as wild cards. An asterisk must be before or after the argument. They may not be embedded within the argument.
- %true% used to indicate that all values for the given rule are valid.
- %false% used to indicate that no values for the given rule are valid.

For example, to register an editor that triggers when a response is issued for:

- Any page with a content type that contains an image, or the path (file name) ends with .jpg or .gif, regardless of the case of the extension:

```
Condition = (content-type=*image* | path~*.jpg | path~*.gif)
```

- Any text/html page that has a response code in the 400's (invalid page):

```
Condition = ((content-type=text/html) & (responseCode=4*))
```

- Any text page (HTML, XML, etc.) that has a response code other than the 400's:

```
Condition = ((content-type=text*) & !(responseCode=4*))
```

9.5.2.5 Packaging the transcoder files

In order to make your transcoder available to Transcoding Publisher administrators, you must package the class files and property file associated with the MEGlet in a JAR file. Use the normal Java class directory hierarchy rules for adding classes that are part of a package to the JAR. Transcoding Publisher requires that each JAR file contain the files associated with only one MEGlet. If you have more than one MEGlet, you will need to create a separate JAR file for each (as with the HelloWorld sample).

When an administrator registers the transcoder with Transcoding Publisher, the Administration Console will unzip the contents of the JAR file and install the MEGlet. As described in 9.5.2.2, "Naming your MEGlet" on page 200, the transcoder's class files will be stored under the <install_path>/addedPlugins directory, where <install_path> indicates the directory where you installed Transcoding Publisher. The property file for the transcoder will be stored under the <install_path>/etc/plugins directory.

To ensure that your transcoder's files are installed according to the hierarchical naming convention you have selected, the contents of your JAR file should be organized a certain way. Suppose you have created a transcoder class called MyMeglet.class and you want to install it in a directory structure according to your company name. This will separate your company's transcoder from others and display it in the Administration

Console under its own node of the Transcoders tree. Before you create your JAR file, organize your files according to the following structure:

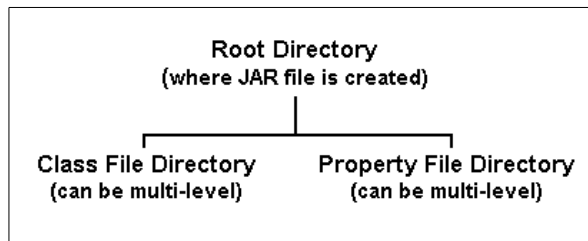


Figure 166. Organizing MEGs structure

For our MyMeglet example, from the current directory, you might create a `com/myCompany/myPackage` directory path and put `MyMeglet.class` in it. Note that because `addedPlugins` is in the classpath, the package for this would be `com.myCompany.myPackage`. For the property file, you might create a `myCompany` directory under the current one. Then you could create the JAR file in the current directory, reflecting this organization by specifying:

```
jar -cvf MyMeglet.jar com/myCompany/myPackage/MyMeglet.class  
myCompany/MyMeglet.prop
```

After registering MyMeglet with the Administration Console, the class file is stored in `addedPlugins/com/myCompany/myPackage/MyMeglet.class`, and the directory path used for the property file is reflected in the folder organization of the Transcoders tree, as shown in the following window fragment:

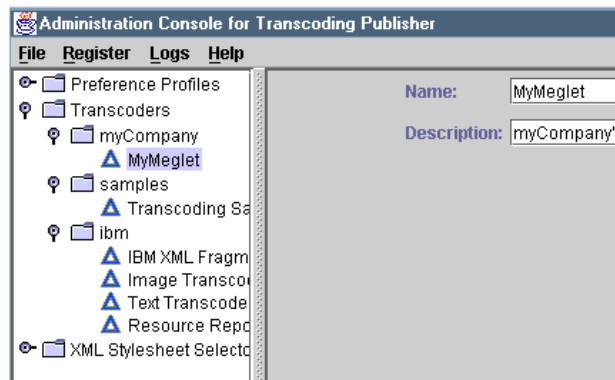


Figure 167. Administration Console - MyMeglet

Note that there is no reason you could not use the same directory path for both the class and property files when building the JAR file. It simply depends on what structure you require to provide an appropriate level of differentiation between your transcoder and those of other developers.

If your transcoder requires additional files, for transcoder-specific information or other data, these should be put in the same directory as the class file and included in the JAR with similar path information.

You can look at the JAR files included with the Toolkit samples for examples of different ways of packaging transcoder files. For more information about creating JAR files or using the Java Archive Tool (the `jar` command) in the JDK, you can refer to The Java Archive (JAR) File Format:

(<http://developer.java.sun.com/developer/Books/JAR/index.html>)

9.5.2.6 Adding a new preference profile

Depending on what your MEGlet is designed to do, you might find that you need to take advantage of Transcoding Publisher's preference profile support. For example, your MEGlet might require an entirely new profile composed of known preferences but with values and settings not supported in existing profiles.

For information about how Transcoding Publisher uses preference profiles and the different ways that you can customize them see 12.3, "Creating preference profiles" on page 268.

9.5.2.7 Registering the MEGlet with Transcoding Publisher

The final step in making a MEGlet ready for use is to make the transcoder files available to Transcoding Publisher. By doing so, you can verify that the MEGlet functions as intended before you distribute it to others for their own use.

To install the MEGlet, you will need to use the Administration Console. The console provides wizards for registering new transcoders and preference profiles and also provides access to Transcoding Publisher's tracing functions. For details on using the Administration Console see 12.2, "Using the Administration Console" on page 261.

Chapter 10. Transcoding wireless applications

In this scenario we will explain configurations, considerations and the basic definitions needed to build net access for wireless devices using IBM WebSphere Transcoding Publisher.

10.1 Introduction

When talking today about wireless communications we here assume that all wireless devices are using WAP (Wireless Application Protocol). WAP is the widely accepted and leading standard for using applications over wireless networks. Using the WAP protocol, Internet and intranet data can be delivered to digital phones and other wireless devices like pagers, smartphones, communicators etc. in a secure manner. The protocol is designed to be used with networks like CDPD, CDMA, GSM, PDC, PHS, TDMA, FLEX, ReFLEX, iDEN, TETRA, DECT, DataTAC, and Mobitex. The implementation of the solution is basically independent on the gateway provider.

The fundamental limitations on wireless solutions are the limitations that devices will have like:

- CPU power
- Memory (RAM and ROM)
- Power
- Display size
- Input devices

These handheld devices have to be operated with a battery and the battery should operate a long time. Also, to be useful, these devices should be as small as they can and also be lightweight. These factors will limit the bandwidth which can be used, the size of the screen and the input device. The WAP protocol is designed to standardize the way data can be delivered independent of the carrier or the device. Nokia, Ericsson, Motorola and Phone.com founded the WAP Forum and since then, the forum has grown to over 100 members.

More information on the WAP protocol and the members who are building the WAP Forum can be found at <http://www.wapforum.org>.

The WAP Forum has defined several standards for WAP and the current level of WAP is now 1.2. Most of the current implementations are 1.1 compliant.

Next, we describe the WAP specification because we will refer in this chapter to WAP several times. The key elements of the WAP specification are:

1. WAP programming model
2. A markup language (WML and WMLScript)
3. Specification of the microbrowser
4. Protocol stack
5. A framework for Wireless Telephony Applications (WTA)

10.1.1 WAP programming model

The programming model is much like the WWW programming model.

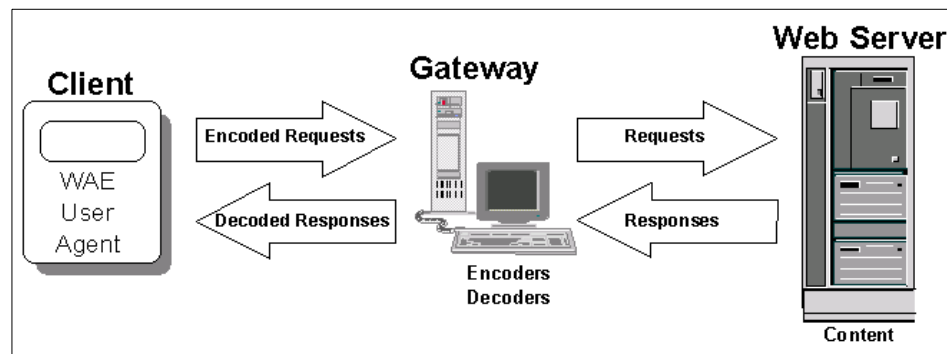


Figure 168. The WAP programming model

The gateway will act like a proxy, getting the content from different sources and translating the TCP/IP protocol to the WAP protocol. Also the HTML markup language is not suitable for mobile devices so the gateway should translate the content to WML and WMLScripts (Wireless Markup Language).

WML and WMLScript are defined for handheld devices which do not have a standard keyboard or mouse for input and the screen size is also very small. The WML consists of small units called *cards*. The basic idea is to make small units of work which require user input and an output suitable for wireless devices. Users can navigate between these cards by using scrolling keys or an equivalent. The gateway will keep track of the cards and make the cardsize suitable for the device. The gateway will use standard HTTP 1.1 requests to get the content from traditional Web servers.

10.1.2 Microbrowser

Handheld devices have scaled down browsers of their own called microbrowsers and they can vary from device to device. The microbrowser will control the user interface on the handheld device. It works like any Web browser. Microbrowsers have the ability to interpret WML and WMLScripts and show content to the user.

10.1.3 Lightweight protocol stack

Since in mobile connections we do not share the communications between many users like TCP/IP, the protocol between the gateway and the device can be much lighter.

The protocol is defined so that the capacity of the connection is efficiently used. Mobile devices do not have to have a TCP/IP stack. In the protocol stack there is a secure layer which is based on the industry-standard Transport Layer Security (TLS) protocol, the former Secure Socket Layer (SSL).

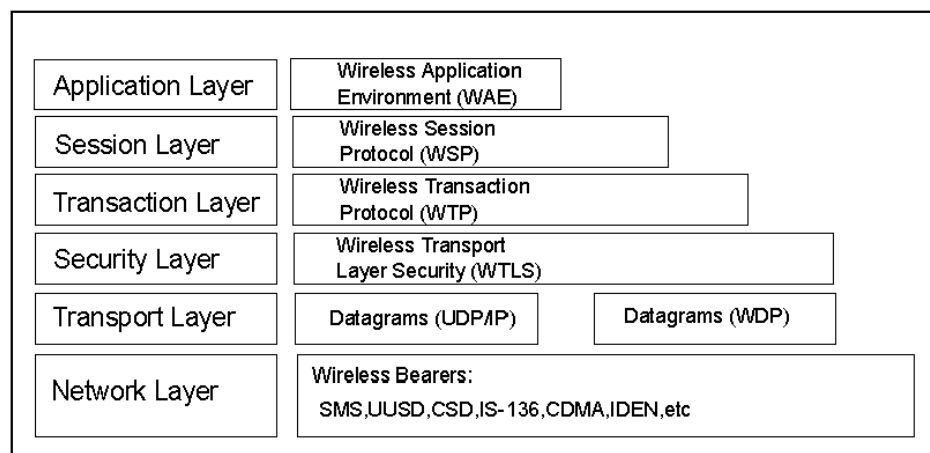


Figure 169. The WAP protocol stack

10.1.4 Wireless telephony applications framework

The framework offers access to telephone functionality like call control, phone book access and messaging within WMLScript applets. This means you can build applications that can make calls automatically etc.

10.2 Using Transcoding Publisher in wireless communication

In this part, we will describe a scenario where we will use IBM WebSphere Transcoding Publisher in co-operation with a WAP gateway. The WAP gateway takes care of the communications for the mobile device. The type of network itself doesn't affect in any way how IBM WebSphere Transcoding Publisher processes requests. So it could be a GSM network or any other network between the gateway and the device.

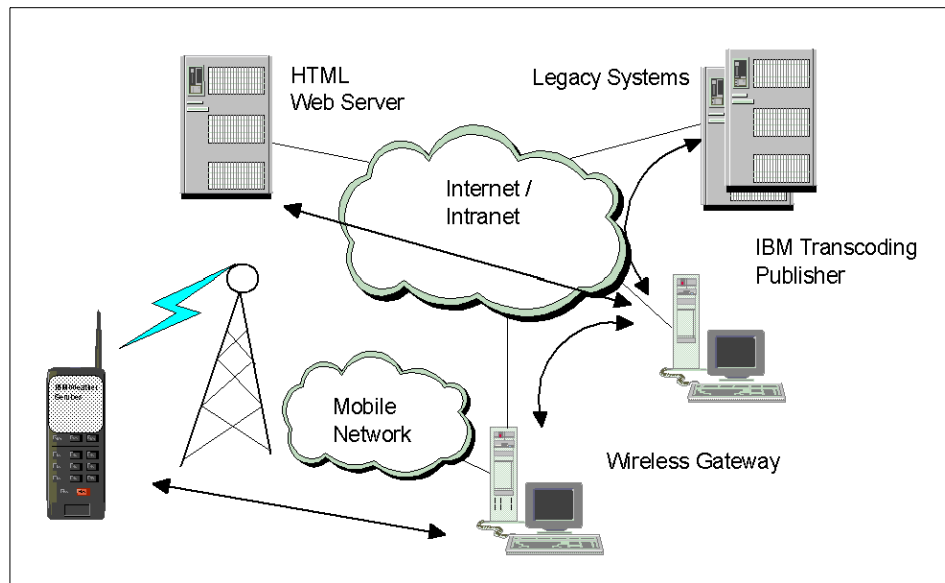


Figure 170. Using IBM WebSphere Transcoding Publisher for wireless networks

In our examples we will explain how to configure IBM WebSphere Transcoding Publisher and we will use a Nokia emulator acting as a WAP phone. Our lab will not use a WAP gateway since the WAP protocol is not offered yet by teleoperators in this location. Here we use TCP/IP as a communication protocol instead. The configuration for IBM WebSphere Transcoding Publisher will still be as used with a WAP gateway. The WAP gateway will be used in our scenario for the protocol for the device like a GSM phone.

We can use IBM Transcoding Publisher in several ways when we are building a solution as depicted in Figure 170.

1. JavaBeans
2. Filters in WebSphere

3. Servlet in gateway
4. Used as a proxy
5. Used as a proxy with cache

10.2.0.1 IBM WebSphere Transcoding Publisher used as JavaBeans

The WAP gateway can be built with the option to build applications in the gateway using JavaBeans serving as transcoders under the gateway. This is highly dependent on the capabilities on the WAP gateway solution. Some gateways have this capability. All security options are available, depending on the capabilities on the WAP gateway. WTP JavaBeans are described in Chapter 8, “Transcoding with JavaBeans” on page 121.

10.2.0.2 Transcoding Publisher as filters in WebSphere

Here, WebSphere will serve the WAP gateway by delivering either HTML or WML output. If HTML is used then the HTML-to-WML translation has to happen in the WAP gateway. WebSphere applications can also use IBM WebSphere Transcoding Publisher to produce WML which is then used in the WAP gateway to encode the data to the WAP device. All security options are available, depending on the capabilities on the WAP gateway. Using IBM WebSphere Transcoding Publisher as a filter is described in Chapter 7, “Transcoding with WAS filters” on page 93.

10.2.0.3 IBM WebSphere Transcoding Publisher used as a proxy

This is the most common way to implement IBM WebSphere Transcoding Publisher. We will use this in our example. IBM WebSphere Transcoding Publisher will act as a proxy, getting data from various sources and it will transcode the data to be suitable for WAP devices. In this model the security is between the WAP gateway and the device.

10.2.0.4 Transcoding Publisher used as a proxy with cache

This solution is similar to the previous scenario with the exception that now transcoded results will be saved in an external cache. How long those results are usable is based on the source. This might overlap with some of the gateways, because they might have their own caches. If this solution is used a consideration has to be made as to how the cache would give best performance. In this model the security is between the WAP gateway and the device, not between IBM WebSphere Transcoding Publisher and the WAP gateway or between IBM WebSphere Transcoding Publisher and the source.

10.3 Wireless network scenario

The basic definitions for wireless networks in IBM WebSphere Transcoding Publisher are the Device and Network Preference Profiles as illustrated in Figure 171.

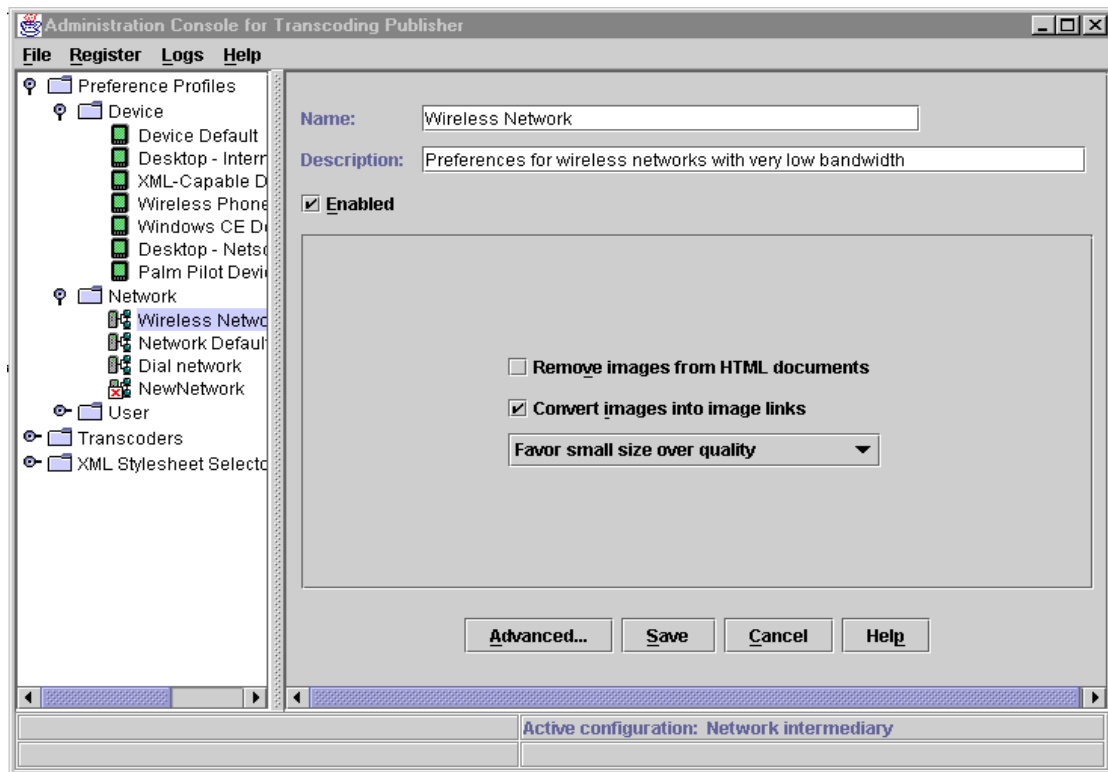


Figure 171. Network preference profile for wireless network

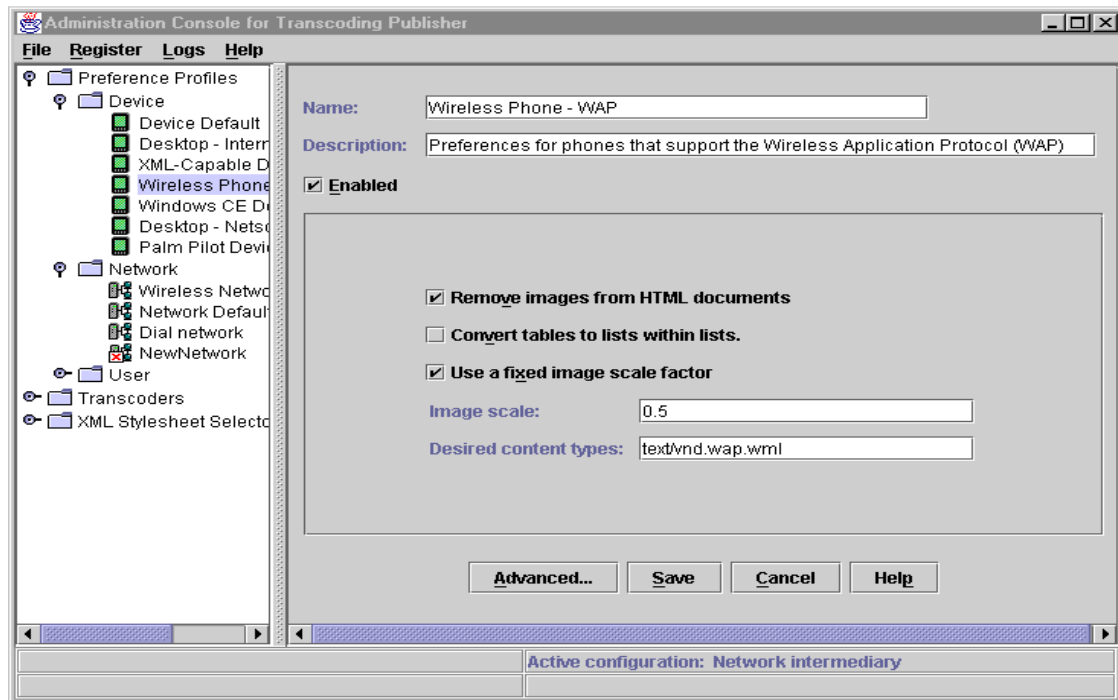


Figure 172. Wireless phone device preference

Figure 172 shows the values that can be changed using the Administration Console.

10.3.1 Components used in this scenario

There are several components used in this scenario. We explain the most important ones in some detail.

10.3.1.1 Text Transcoder

IBM WebSphere Transcoding Publisher provides extended function for transcoding data from HTML to Wireless Markup Language (WML). This capability is used when modifying an HTML browser-based application for display on a client device which will use the WAP protocol. This transcoder will do the generation of a generic WML document from an incoming HTML document. Here is a sample where the transcoding has done using the Transform Tool from the toolkit.

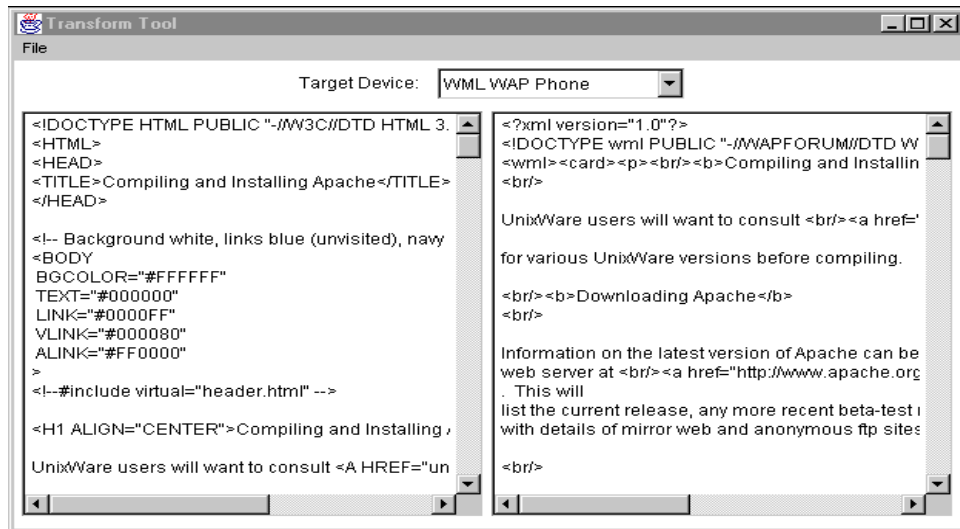


Figure 173. HTML transcoded into WML

10.3.1.2 Text clippers

There exists the ability to apply customized transcoders, called text clippers, that can extract specific portions of the WML document to be displayed on the device. A text clipper is closely associated with the document it is processing and is especially useful for pulling only necessary information from an otherwise large document. IBM WebSphere Transcoding Publisher toolkit includes two variations on a sample text clipper, the IBM Stock Clipper, to demonstrate how text clipping can be done:

- IBMStockClipper.jar

This version uses "text clipping" against a textual version of the original HTML document after it has been transcoded into WML.

- IBMStockClipperDom.jar

This version, the IBMStockClipperDOM, operates on DOM. DOM stands for Document Object Model. This simply means that the WML document is passed within a structured layout where each of the formatting elements is passed as an object which can be more easily manipulated.

The goal of the document clipping process is to generate a generic WML document from an incoming HTML document and then allow text clippers (special WBI response editors) to extract the needed portions of the document for display on the WML browser. The generated WML document that IBM WebSphere Transcoding Publisher produces is available both in its

text form and as a Document Object Model (DOM). The DOM defines the logical structure of the document and the way the document can be accessed and manipulated.

After the generic WML document has been generated, there are three approaches to extract needed information from the WML document:

- Operate on the text form of the WML document with Java code (for example, with a response editor).
- Operate on the WML DOM with Java code (for example, with a response editor).
- Operate on the WML DOM with stylesheets.

Because information on how to apply style sheets to handle this type of problem is more readily available, the text clipper samples provided with IBM WebSphere Transcoding Publisher address the first two approaches described above. Generally speaking, operating on the WML DOM is probably preferable to operating on the text form of the document, assuming that a useful set of functions to operate on the DOM are available.

However, we do not want to make too many assumptions about the type or variability of the source content. In some cases, parsing the text form of the WML to extract the final document may be quite feasible. For that reason, the text clipper samples that IBM provides demonstrate how to work with either the text form of the WML or the WML DOM.

10.3.1.3 WML deck fragmentation

Converting an HTML page to WML is very likely to result in a WML document that exceeds the maximum deck size (storage capacity) of the device. The same problem may be encountered with WML content if the content generator was unaware of the specific limits of the device being used. Exceeding the storage capacity of the device means the page cannot be viewed on that device. WML deck fragmentation solves this problem by splitting a single oversized WML deck into multiple smaller units, each one smaller than the WML maximum deck size limitation.

In addition to splitting up the deck into smaller chunks, the WML fragmentor must add links to each of the generated pieces to allow navigation from one piece to the next (and previous). "Continue..." moves to the next fragment; "Back" moves to the previous one (the first fragment has no "Back" and the last no "Continue..."). The maximum WML deck size is a property of the device, so the WML fragmentor uses the device information retrieved from the device preference profile to determine the maximum deck size value.

If this value needs to be modified we have to define a new Device profile and checkmark the Configurable box. In the default definition this is not configurable.

Profile preference	Include?	Configurable?	Default value
Desired content types:	<input type="checkbox"/>	<input type="checkbox"/>	[text/html, text/xml]
Cascading stylesheets are supported.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Cascading stylesheets are supported.
WMLMaximumDeckSize	<input checked="" type="checkbox"/>	<input type="checkbox"/>	1440
Use a fixed image scale factor	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> Use a fixed image scale factor
Image types that are not supported	<input type="checkbox"/>	<input type="checkbox"/>	[]
Compress the size of HTML text files	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> Compress the size of HTML text files
HTML frames are supported.	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> HTML frames are supported.
Image types that are supported	<input type="checkbox"/>	<input type="checkbox"/>	[gif, jpg]
File extensions that are not supported	<input type="checkbox"/>	<input type="checkbox"/>	[]
Java Script is supported.	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> Java Script is supported.
Device screen capability	<input type="checkbox"/>	<input type="checkbox"/>	High
Image maps are supported.	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> Image maps are supported.
Java applets are supported.	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> Java applets are supported.
Remove images from HTML documents	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> Remove images from HTML documents
Desired document type definitions:	<input type="checkbox"/>	<input type="checkbox"/>	[]
Objects supported	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> Objects supported
Image scale:	<input type="checkbox"/>	<input type="checkbox"/>	0.8
Convert tables to lists within lists	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> Convert tables to lists within lists

Figure 174. Definition for the maximum deck size

The WML fragmentation engine stores non-first fragments in a general-purpose "resource repository". The resource repository appears as a transcoder in the Administration Console. Making the resource repository general purpose will allow for reuse by other components needing a similar service in the future. But for now, only the WML fragmentation engine uses the resource repository.

Because the WML fragmentor needs the resource repository to save fragments for later retrieval, the resource repository should NOT be disabled if WML fragmentation is being used. The Administration Console does not enforce this because it is unaware of the dependence. If the WML fragmentor is disabled, the resource repository should also be disabled (this will improve performance).

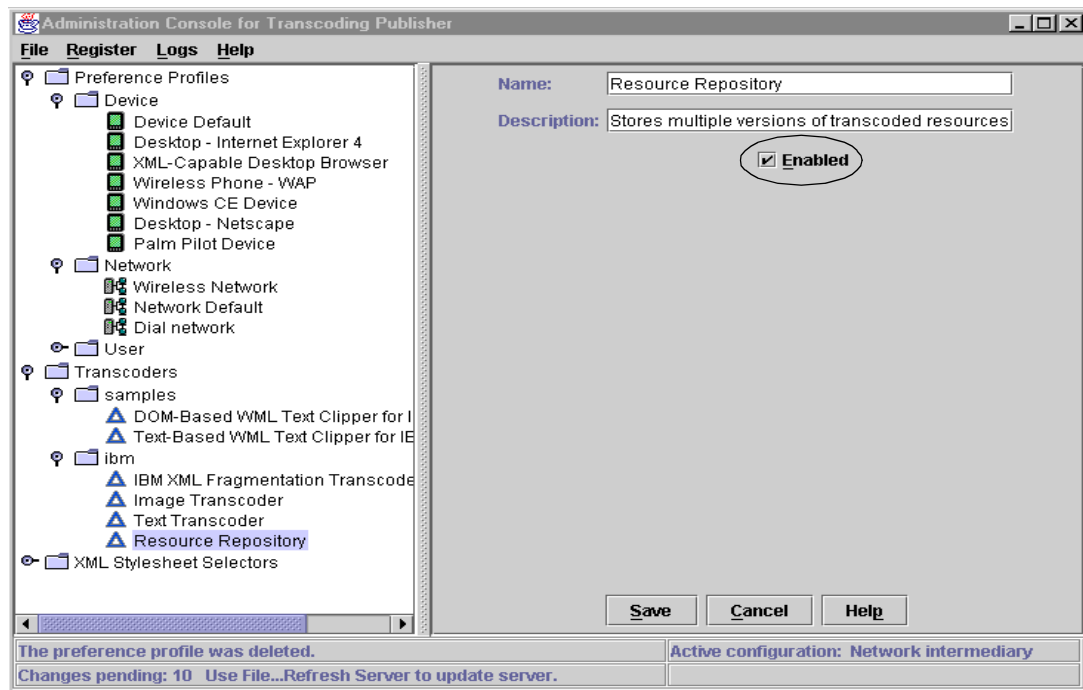


Figure 175. Resource repository in Administration Console

Figure 176 shows an example of WML Fragmentation.

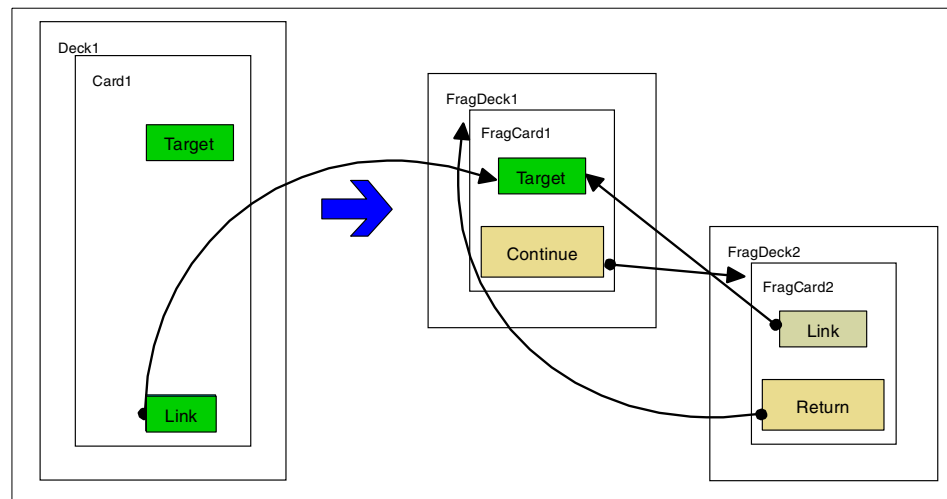


Figure 176. Sample WML fragmentation

A single oversized WML deck is fragmented into two smaller pieces. "Continue..." and "Return" links are inserted into the fragments to allow for navigation between the fragments. Also, any intra-deck links in the original deck are updated to point to the target in whatever deck/card it gets placed in.

Sometimes content cannot be fragmented into small enough pieces:

- Can fragment on <p> and
 tags.
- Long paragraphs with no breaks can cause problems.

10.3.2 Text clippers and the fragmentation transcoder

Depending on the size of the document that your text clipper produces, IBM WebSphere Transcoding Publisher fragmentation transcoder can be used to subdivide the WML output into chunks that the requesting device can handle. The text clippers will run before the fragmentation transcoder, since the clippers are assigned a higher priority in their respective property files. This ensures that the text clipper receives an unfragmented document on which to work.

If the text clipper reduces the document down to a size that fits within the constraints specified for the requesting device, the fragmentation transcoder will not alter the document. Otherwise, the document will be fragmented into decks, which will be sent to the device in sequence.

Fragmenting of WML documents is performed automatically by IBM WebSphere Transcoding Publisher as long as the fragmentation transcoder is enabled. There is no additional configuration required on the developer's part to use the fragmentation transcoder with custom text clippers.

10.3.3 Generic HTML-to-WML transcoding

As mentioned above, the text transcoder supplied with IBM WebSphere Transcoding Publisher is capable of converting HTML data into WML form. This section describes the transformations performed in generating a WML document (and DOM) from the original HTML document, as well as which tags are deleted or replaced.

The following HTML tags are transformed:

Tag Name	Transformation Description
Anchor (A)	If the BASE attribute was stored from the previous Head element (see description of Head element below), a relative path is prefixed with the base value. If the TARGET attribute is specified on the Anchor element it is removed. Also, if no HREF attribute is specified the entire element is removed.
Body (BODY)	The BODY element should be the first element found under the HTML element and is replaced with a WML card element with a P element as its child. All the children of the BODY element are moved under the created P element.
Bold (B)	If the Bold (B) element is a child of an Anchor element, the children are removed from the node and put in its place. This is because in WML an Anchor is not allowed to have a Bold element as a child. Moving up the children in this way has the effect in the output of looking like the outer tag (in this case the and). If the Bold element has a FORM HTML element as a child, the children are likewise moved up in place of the Bold element.
Break (BR)	If the Break element is a child of a WML or TD element, it is removed. Otherwise, only the attributes are removed.
Definition List (DL, DT, DD)	The children are moved up in place of the Definition List element. Break elements are inserted before the children that were moved up, if they do not automatically cause a break.
Font (FONT)	If the FONT element is a child of an Anchor (A) or Strong (STRONG) node, the children are moved up in place of the element. Otherwise, the FONT element is turned into a STRONG element.
Form (FORM)	The FORM element usually contains INPUT and SELECT elements. The FORM element is traversed and if a POST or GET action is found and an INPUT with a VALUE specifier is found, a WML DO element (with an equivalent INPUT element) is created. Similarly, an equivalent SELECT element is created if the HTML SELECT has a VALUE specifier. The following TYPE attributes are currently supported: "submit", "password", "text", "radio", and "checkbox". If TYPE is not specified, "text" is assumed.
Head (HEAD)	If the HEAD element specifies the BASE attribute, this value is saved in the MegContext object for later use in processing the Anchor element.
Heading (H1...H6)	The Heading element is replaced with a Bold element. In addition, Break elements are created and placed before and after the heading.

Tag Name	Transformation Description
HTML (HTML)	The HTML element is replaced with a WML element, and the children of the HTML element are moved underneath the new WML element.
Image (IMG)	The Image element is removed if there is no alternate text (ALT attribute) specified. Otherwise, if the disposelimages preference is true, the image is replaced with its alternate text when inside an Anchor node and removed otherwise. All BORDER and USEMAP attributes are also removed.
List (UL, OL, LI)	The children are moved up in place of the List element. Break elements are inserted before the children that were moved up, if they do not automatically cause a break.
Paragraph (P)	If the Paragraph element is not the child of the newly created CARD element, Break elements are inserted before and after the children of the P element, and the P tags themselves are removed.
Small (SMALL)	If the Small element is a child of an Anchor (A) element, the Small element is moved up in the Anchor's place. Otherwise, the node is unchanged.
Table (TABLE)	<p>The Table element is processed by first moving up the elements of nested tables, which are not allowed in WML. Table Header (TH) elements, which are not supported in WML, are replaced by TD elements. Those tables that are not nested have the elements moved up in place of the table when:</p> <ul style="list-style-type: none"> • They contain elements that are not allowed in WML tables. • They have only one column. • They have more than 3 columns.

The following HTML tags are deleted:

Address (ADDRESS)	Comment (COMMENT)	Meta (META)
Applet (APPLET)	Horizontal Row (HR)	Script (SCRIPT)
Area (AREA)	Layer (LAYER)	Span (SPAN)
Base Font (BASEFONT)	Link (LINK)	Style (STYLE)
Background Sound (BG SOUND)	Map (MAP)	Title (TITLE)

HTML tags replaced by their children are:

Block Quote (BLOCKQUOTE)	Frame (FRAME)	Preformatted (PRE)
Center (CENTER)	Frameset (FRAMESET)	Quoting (Q)
Cite (CITE)	Floating Frame (IFRAME)	Sample (SAMP)
Code (CODE)	Insert (INS)	Strike (S)
Delete (DEL)	Keyboard (KBD)	Strike (STRIKE)
Division (DIV)	Legend (LEGEND)	Typewriter Text (TT)
Fieldset (FIELDSET)	No Frames (NOFRAMES)	Underline (U)

10.3.4 Configuration of the scenario

Here we go through a basic example of the IBM WebSphere Transcoding Publisher used with WAP device. In our example we will use as a client a workstation with Nokia WAP Toolkit 1.1 which provides us a WAP device. The overall picture of the configuration is shown in Figure 177.

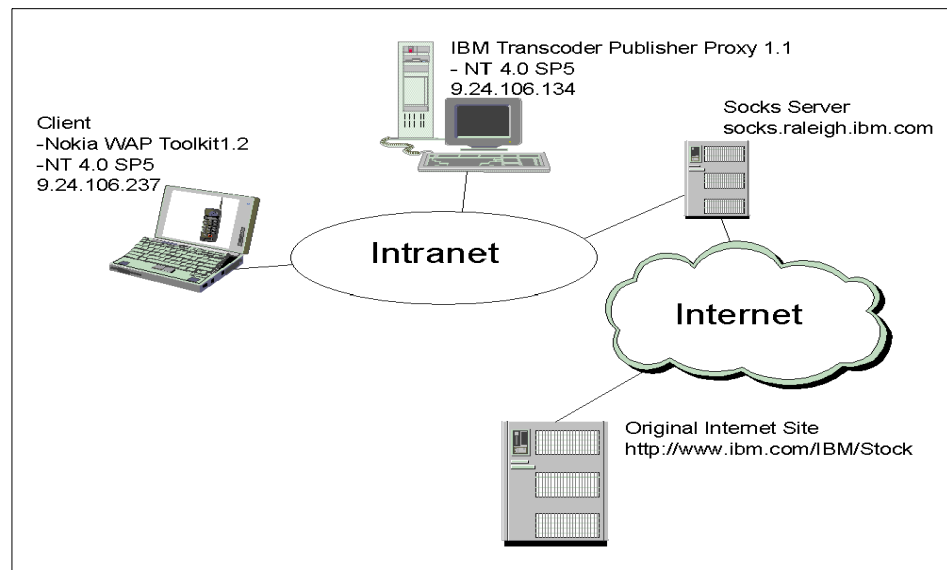


Figure 177. Overall configuration of the WAP device scenario

To configure this environment we first have to configure the IBM WebSphere Transcoding Publisher to access the Internet. In our example we are using SOCKS server: *socks.raleigh.ibm.com*.

To configure the server, from the Administration Console click **File-->Settings**
--> Firewall.

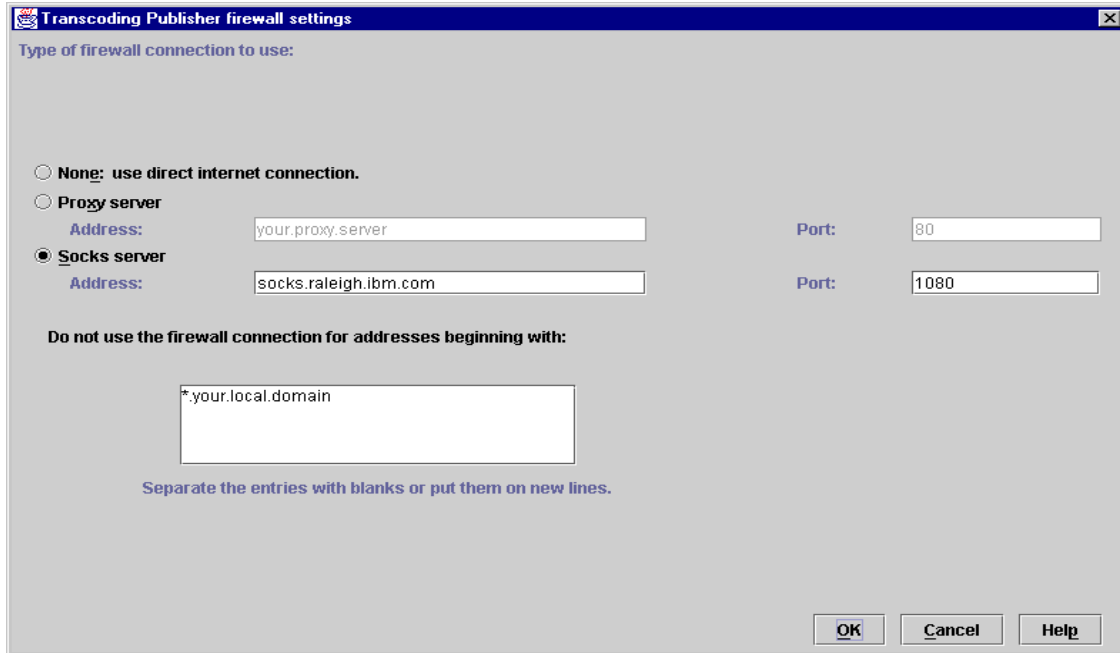
The image shows a dialog box titled "Transcoding Publisher firewall settings". It has a blue title bar with a close button. The main area is light gray. At the top, it says "Type of firewall connection to use:". Below this are three radio buttons: "None: use direct internet connection.", "Proxy server", and "Socks server". The "Socks server" radio button is selected. To the right of the "Proxy server" and "Socks server" options are two sets of input fields. For "Proxy server", there is an "Address:" field with the text "your.proxy.server" and a "Port:" field with the value "80". For "Socks server", there is an "Address:" field with the text "socks.raleigh.ibm.com" and a "Port:" field with the value "1080". Below these fields, there is a section titled "Do not use the firewall connection for addresses beginning with:" followed by a text input field containing "*your.local.domain". Below this field, there is a note: "Separate the entries with blanks or put them on new lines." At the bottom right of the dialog box are three buttons: "OK", "Cancel", and "Help".

Figure 178. Configuring the SOCKS server

Select the **Socks server** radio button and enter the address for the SOCKS server *socks.raleigh.ibm.com* and also enter the port number (unless you are using default values) in the port field. We used port number 1080 (default).

Next, if not using default values, we have to configure the port for the Mobile network. In our case we use the WAP emulator as WML browser and we will use the port defined to Wireless Networks so we configure the port for the proxy:

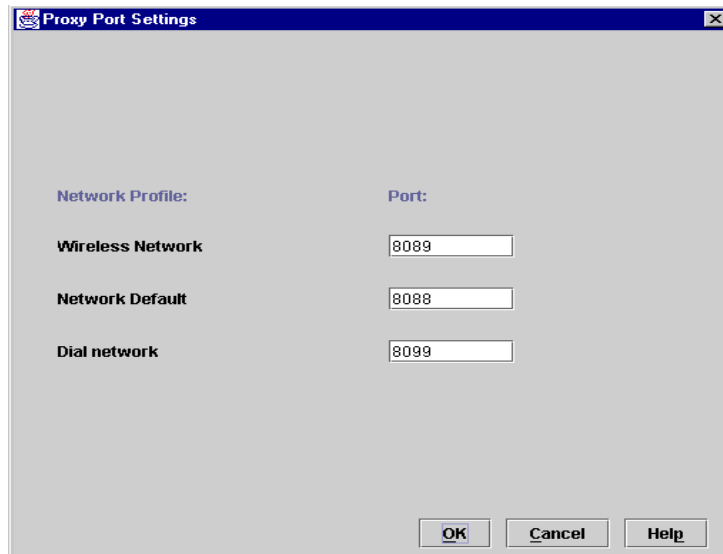


Figure 179. Wireless network port definition

To configure the port from Administration Console select **File -->Settings -->Proxy Port**. In our case we will be using the default port 8089. This will be the port we have to define in our WAP device.

For the Wireless Phone - WAP device profile we will accept the defaults:

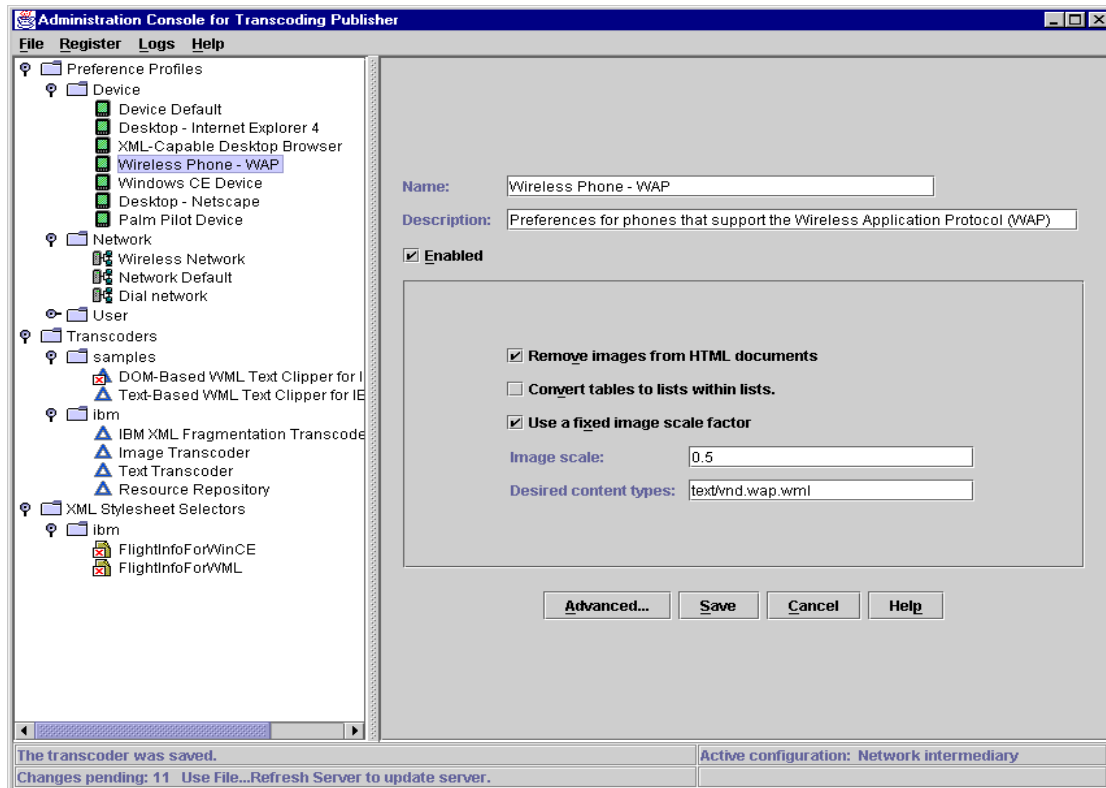


Figure 180. Wireless phone - WAP definitions

From the Administration Console select from the tree view **Preference Profiles --> Device --> Wireless Device - WAP**.

Ensure that the Enabled tick box is checked.

The default configuration has the following configurable options:

- Remove images from HTML documents, checked.
- Convert tables to lists within lists, not checked.
- Use a fixed image scale factor. checked
- Image Scale: 0.5.
- Desired content types: text/vnd.wap.wml

If any stylesheet selectors are needed we would configure them using the Advanced button.

Next we configure the Wireless network preference profile.

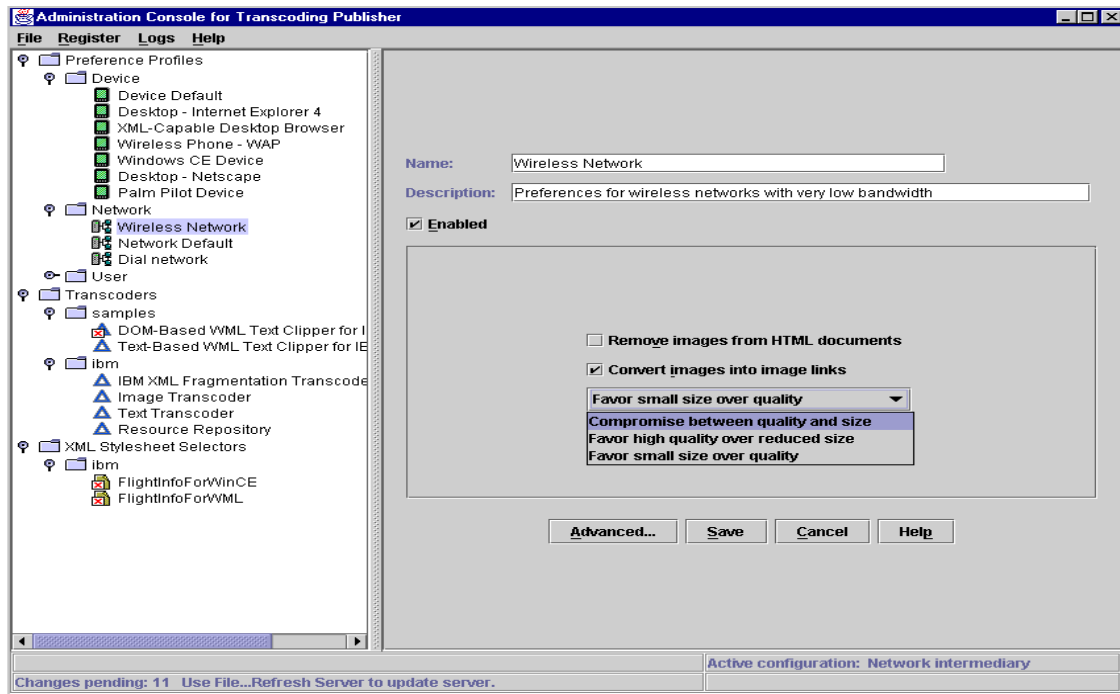


Figure 181. Wireless network preference profile

From the Administration Console select from the tree view **Preference Profiles --> Network --> Wireless Network**.

Ensure that the Enabled tick box is checked.

The default configuration has the following configurable options:

- Remove images from HTML documents, not checked.
- Convert images to links, checked.
- Pull-down list:
 - Favor small size over quality selected.
 - Compromise between quality and size.
 - Favor high quality over reduced size.

If any stylesheet selectors are needed we would configure them using the Advanced button.

Now we have completed the configuration of the IBM WebSphere Transcoding Publisher.

Next we have to configure the WAP device. This is done using the Nokia WAP Toolkit.

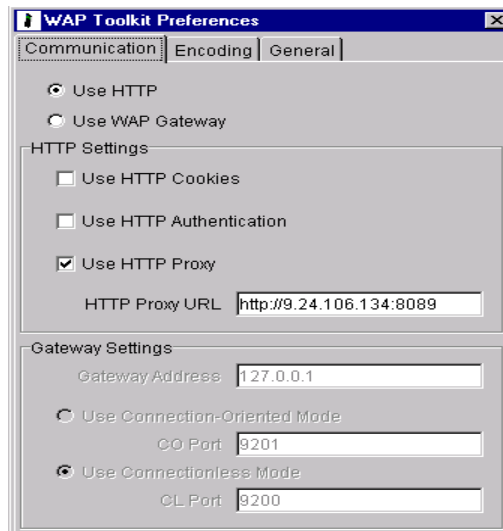


Figure 182. Configuring the WAP device

In the configuration we used an HTTP proxy. The URL for this configuration is the address of our IBM WebSphere Transcoding Publisher proxy and the port we are going to use. In our scenario this will be `http://9.24.106.134:8089`. Now we load the WML file in the WAP emulator as shown in Figure 183.

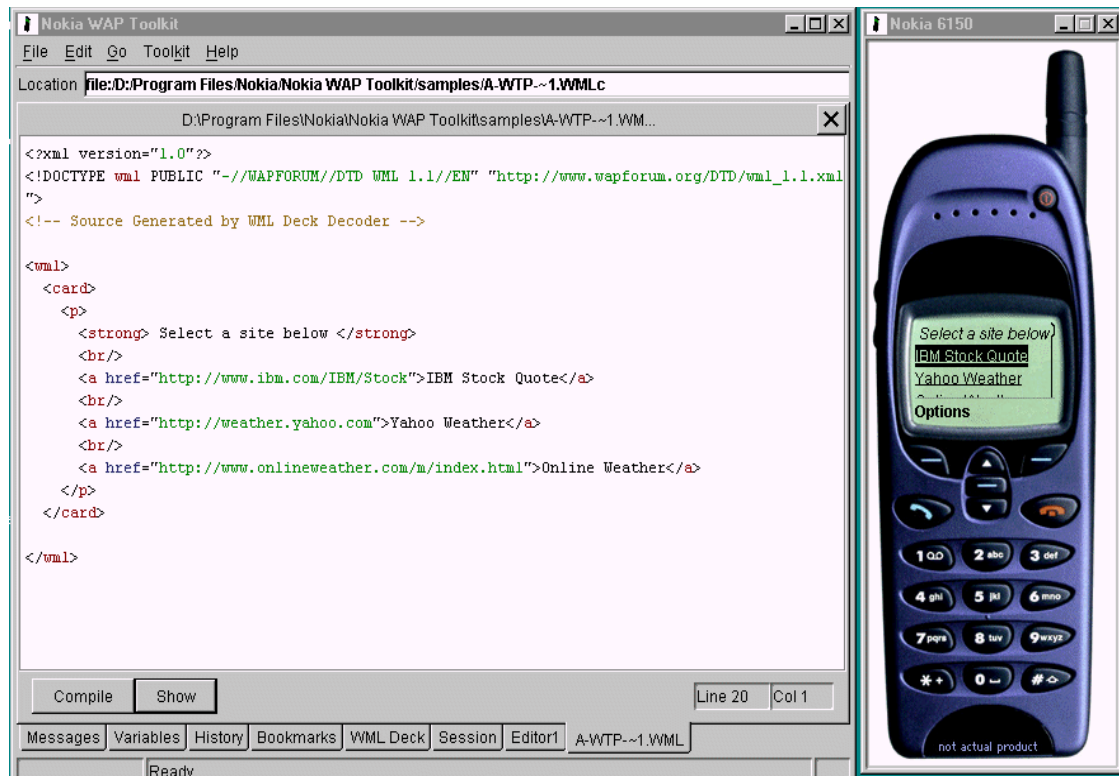


Figure 183. Loading the starting page in WAP emulator

Now we have completed the configuration and we have loaded a starting page to the WAP emulator from the client machine disk. Next, we use the emulator to show us `http://www.ibm.com/IBM/Stock`. Seen from a normal browser this page looks as shown in Figure 184.

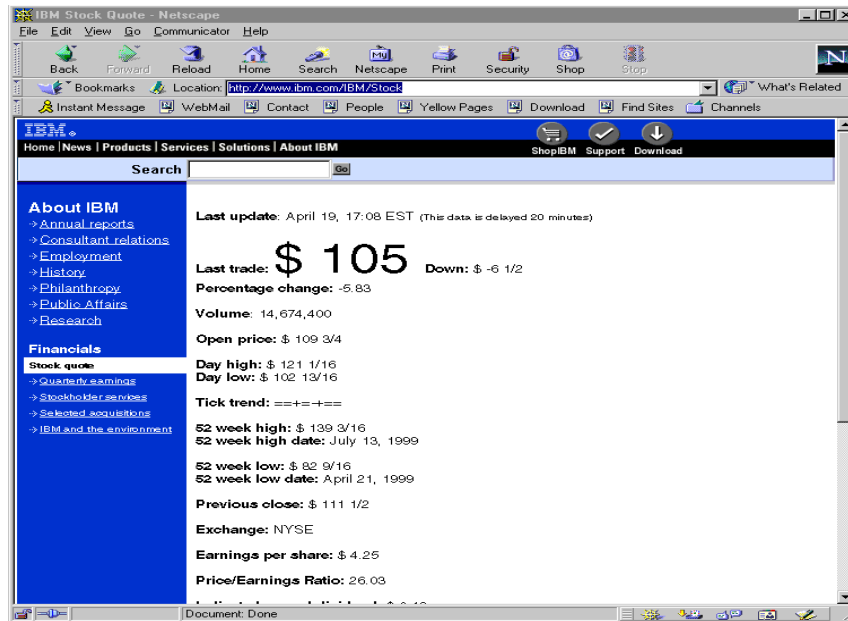


Figure 184. Netscape <http://www.ibm.com/IBM/Stock>

From the WAP device it looks as illustrated in Figure 185.



Figure 185. WAP device <http://www.ibm.com/IBM/Stock/>

In Figure 186 we can see in the Request Viewer tool how IBM WebSphere Transcoding Publisher handled the request. For information about the Request Viewer tool see 9.2, “The Request Viewer” on page 166.

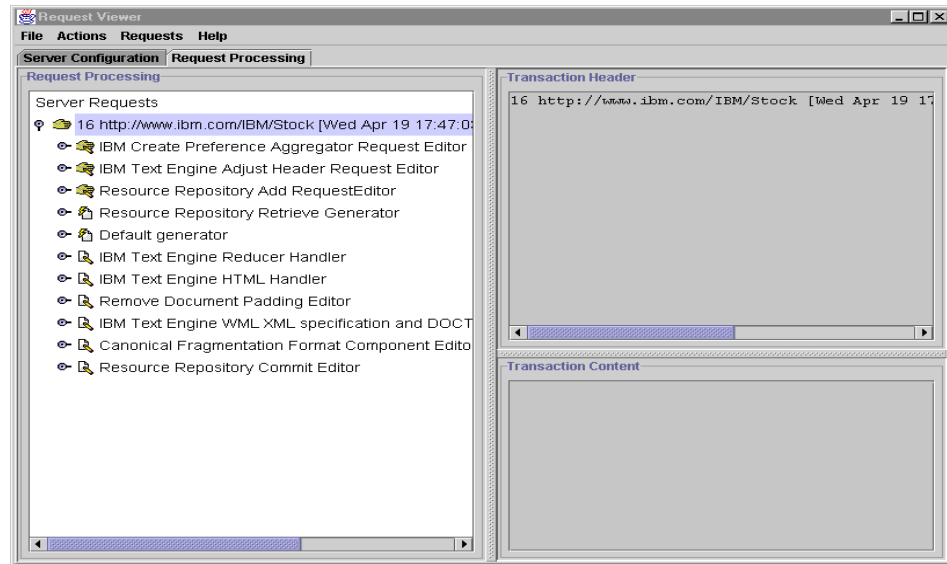


Figure 186. Request Viewer - view of the request to <http://www.ibm.com/IBM/Stock/>

We can also see how the fragmentation has been handled from the Nokia WAP Toolkit:



Figure 187. The deck from <http://www.ibm.com/IBM/Stock>

As we see, the deck consists of only one card.

10.3.4.1 Wireless WAP device used with text clipping

In this scenario we will add a transcoder to IBM WebSphere Transcoding Publisher proxy, but otherwise use the same configuration as before.

To install a transcoder we select from the Administration Console **Register-->Transcoder**.

We have under c:\Program Files\IBMTrans\toolkit\textclippers\IBMStock\ two text clippers and we will now select the IBMStockClipper.jar which is a text based text clipper.

Continue following the windows until you see the Finish button.

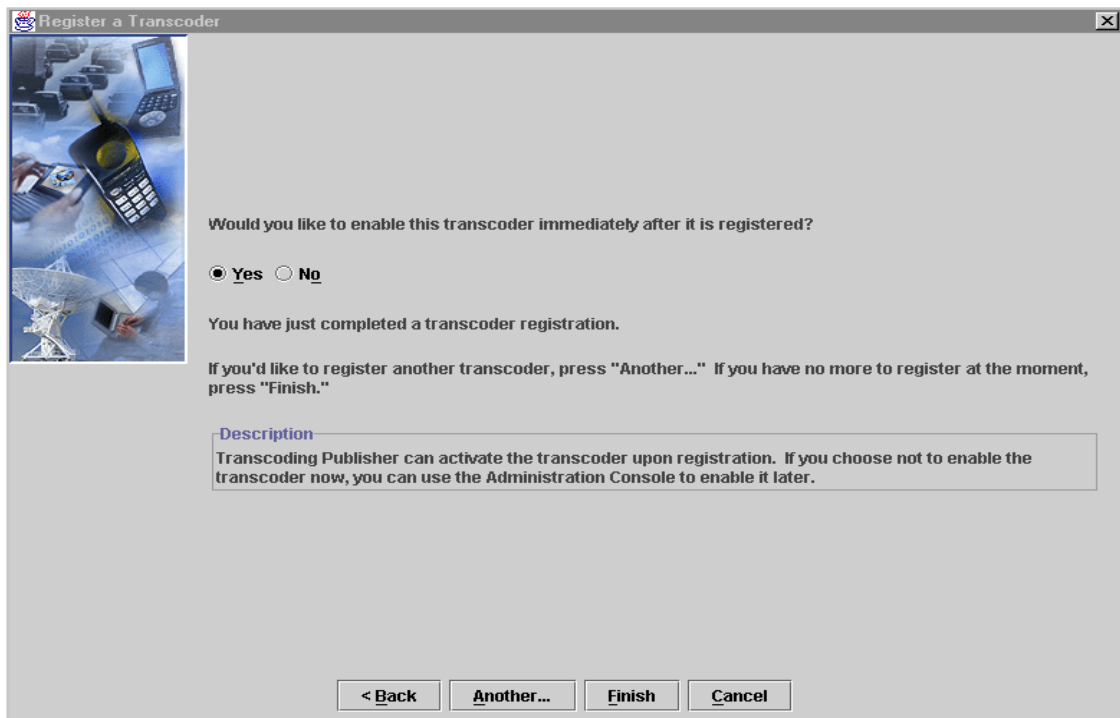


Figure 188. Finish to registering the IBMStockClipper.jar

Select **Yes** to enable the transcoder after registering.

Click **Finish**. Figure 189 shows the text clipper installed in the Administration Console.

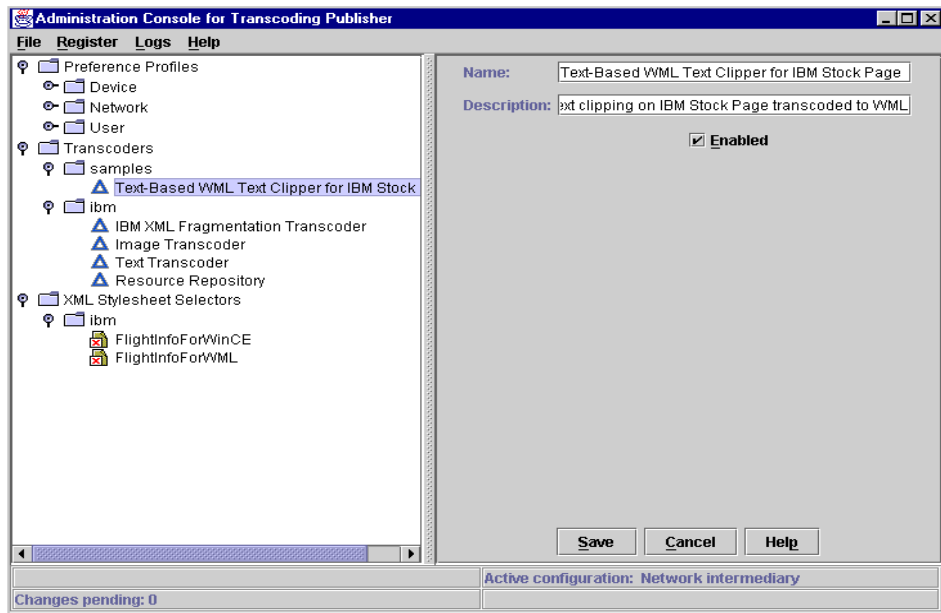


Figure 189. Text-Based Clipper for IBM Stock Page

We can see that the text clipper is enabled.

Next we will access <http://www.ibm.com/IBM/Stock> from the WAP device. The result is shown in the following two figures which show the WML deck and the actual result on a WAP device respectively.

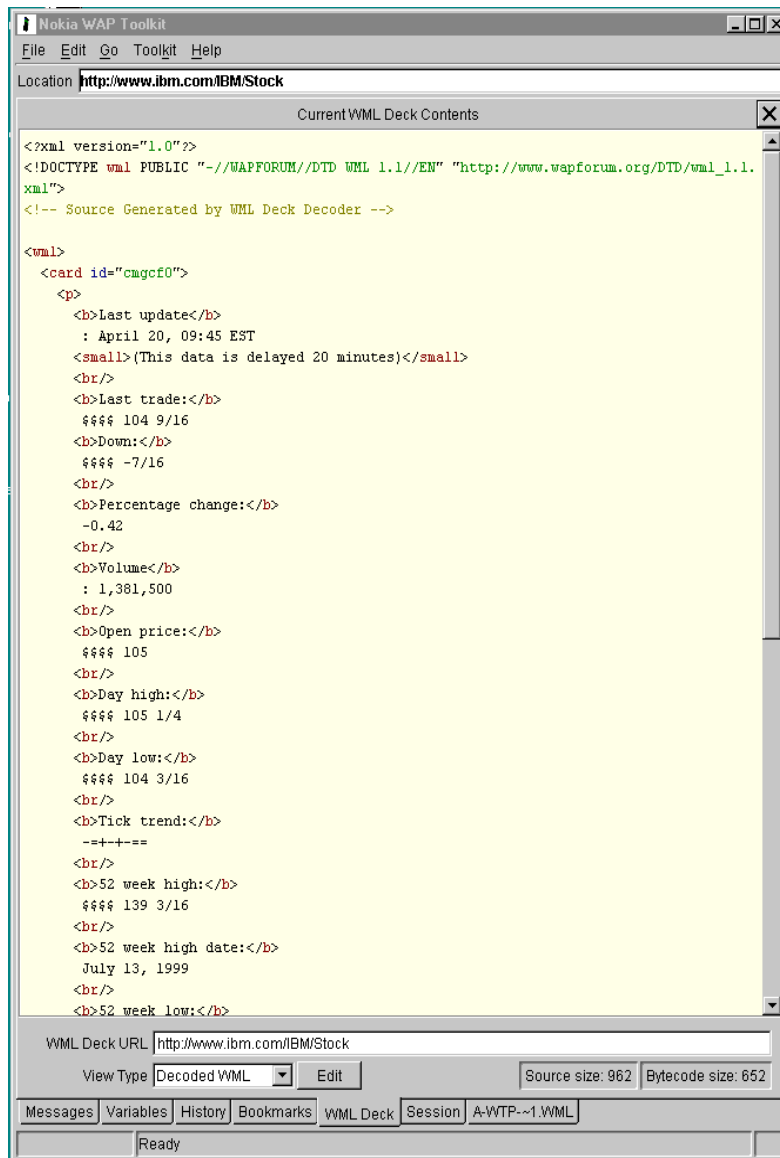


Figure 190. WML deck produced by IBM Transcoding Publisher



Figure 191. Text clipped result

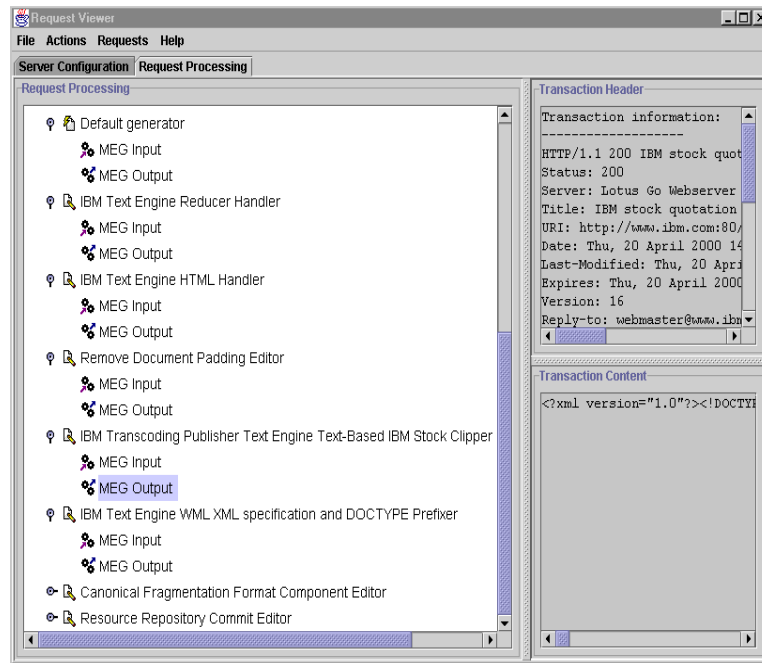


Figure 192. Request Viewer shows the text clipping

Chapter 11. Transcoding Host Publisher application content

Host Publisher applications will typically use JavaServer Pages (JSP) and Integration Objects to access mainframe applications and produce HTML code to be sent to the client devices. Transcoding Publisher technology can also be used in this scenario to allow handheld and other new type devices to access 3270 applications such as CICS and IMS.

This chapter provides a sample scenario using WebSphere Transcoding Publisher (WTP) to transcode Host Publisher applications that access a CICS mainframe application from a WAP device supporting WML.

11.1 Overview

IBM Host Publisher is a key component of IBM's Host Integration software portfolio. Host Publisher V2 provides a Web-to-host solution designed to address the unique characteristics of the Internet. It enables Web integration with existing 3270, 5250, VT, JDBC, and Java host applications, without requiring any changes to those existing applications and using industry-standard HTML Web pages to support end users running non-Java browsers.

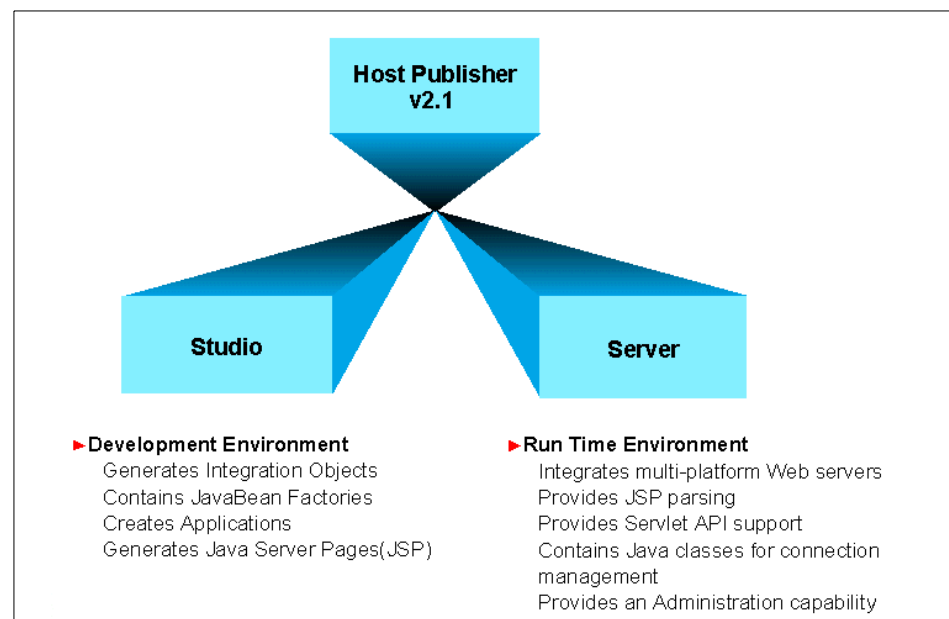


Figure 193. IBM Host Publisher components

Host Publisher offers additional platform support and it provides run-time solutions for S/390, AIX, Windows NT, AS/400 and Solaris operating environments.

As illustrated in Figure 193 on page 235, there are two major components in IBM Host Publisher:

- Host Publisher Studio, which provides easy-to-use tools to assist in the creation of Web-to-host integration projects.
- Host Publisher Server, which consists of IBM WebSphere application server and Host Publisher run-time components. It provides the run-time environment for executing Web applications created with the Host Publisher Studio.

IBM Host Publisher allows you to create and use reusable integration objects. Integration objects created with Host Publisher encapsulate the interaction and data retrieval with host applications. The created integration objects can be reused by other Host Publisher applications as well as Java applications developed outside of Host Publisher.

Other new features included in IBM SecureWay Host Publisher Version 2.1 are:

- Load balancing and hot backup is included with Host Publisher. It is provided by IBM Network Dispatcher, which provides enterprise-class load balancing and hot backup of Host Publisher servers.
- Fully customizable Web pages. It generates fully customizable HTML pages that the application builder can enhance using his or her favorite HTML editor.
- Back-end data sources. It supports the integration of applications from 3270, 5250, Java, and JDBC-compliant databases.
- Connection pools. User-defined connection pools are used during runtime to cache connected, logged on, and ready connections to improve response time.
- Integration object chaining enables you to break the application into steps to provide greater flexibility and performance.
- Integration with other IBM connectors enables integration with other IBM connectors such as the one for MQSeries.
- IBM Host Publisher integrates multiple data sources, including host and database applications, into a single Web page on a client's browser, making host access transparent to the user.

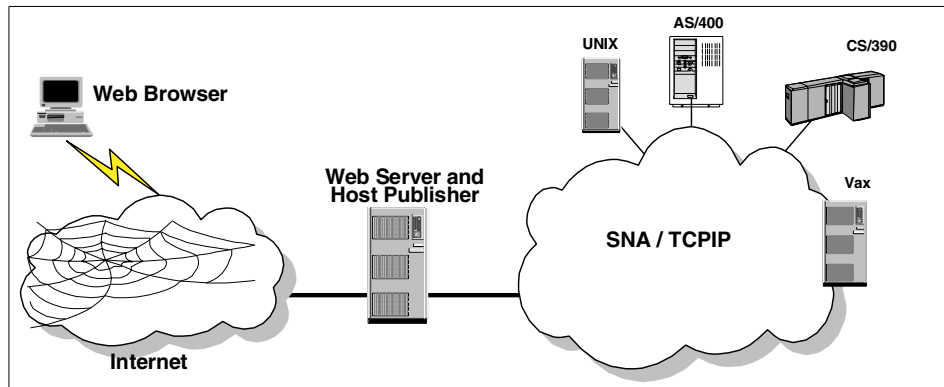


Figure 194. Internet host integration with Host Publisher

IBM Host Publisher is designed for enterprises that need to deliver host information via HTML to standard Web browsers. It is Java-based and supports AIX, OS/390, Windows NT, AS/400 and Sun Solaris operating systems.

Enhancements in Host Publisher Version 2.1 include:

- Support for multiple platforms.
- Ability to create host integration objects that can be reused within Java development tools.
- Inclusion of a run-time version of IBM WebSphere Standard Edition.
- Integration with IBM's connector solutions, such as MQSeries, which significantly extends the breadth of existing applications that can be supported.

Note

For details on how to develop and deploy Host Publisher applications see the IBM redbook *Building Integration Objects with IBM SecureWay Host Publisher Version 2.1*, SG24-5385.

11.2 Transcoding Host Publisher applications

Since Host Publisher applications will typically produce HTML code, they can be transcoded in different ways. For example:

1. Using WebSphere Transcoding Publisher (WTP) running as a proxy server. A caching proxy such as IBM WebSphere Traffic Express (WTE)

- can also be used to cache transcoded pages. See 11.4, “Scenario: transcoding using WTP as a network proxy” on page 256 for details.
2. Using WTP as WebSphere Application Server (WAS) filters. See 11.3, “Scenario: transcoding with WAS filters” on page 238 for details.
 3. Invoking WTP transcoders provided as JavaBeans. See Chapter 8, “Transcoding with JavaBeans” on page 121 for details.

11.3 Scenario: transcoding with WAS filters

In this section we describe how Host Publisher applications can be transcoded when Transcoding Publisher is configured to run as WAS filters.

The following remarks can be made:

- Host Publisher integration objects use the Host Access Class Library (HACL) to communicate with an S/390 or AS/400 system and it uses the Telnet 3270/5250 protocol.
- A TN3270E server can also be used if your mainframe is not configured to support TCP/IP.
- Host Publisher applications require WebSphere Application Server (WAS).
- Host Publisher applications create JavaServer Pages (JSP) that invoke the integration objects.
- Host Publisher integration objects are JavaBeans.
- Integration objects are created with the Host Publisher Studio.
- JSPs are also created with the Host Publisher Studio.

The following restrictions apply when running transcoders as WAS filters in IBM WebSphere Transcoding Publisher Version 1.1:

1. When you enable transcoding for JSPs, WTP filters will be enabled for all JSPs in your server. That is, all JSP applications in your server will be transcoded.
2. Only the device profile is used. That is, the network profile is not used at all.
3. WML fragmentation is not supported in IBM WebSphere Transcoding Publisher Version 1.1 when running as a WAS filter. This support will be included in a future release.

Figure 195 on page 239 illustrates the components when Host Publisher application's content (HTML) is transcoded at the source by running WTP as WebSphere Application Server (WAS) filters.

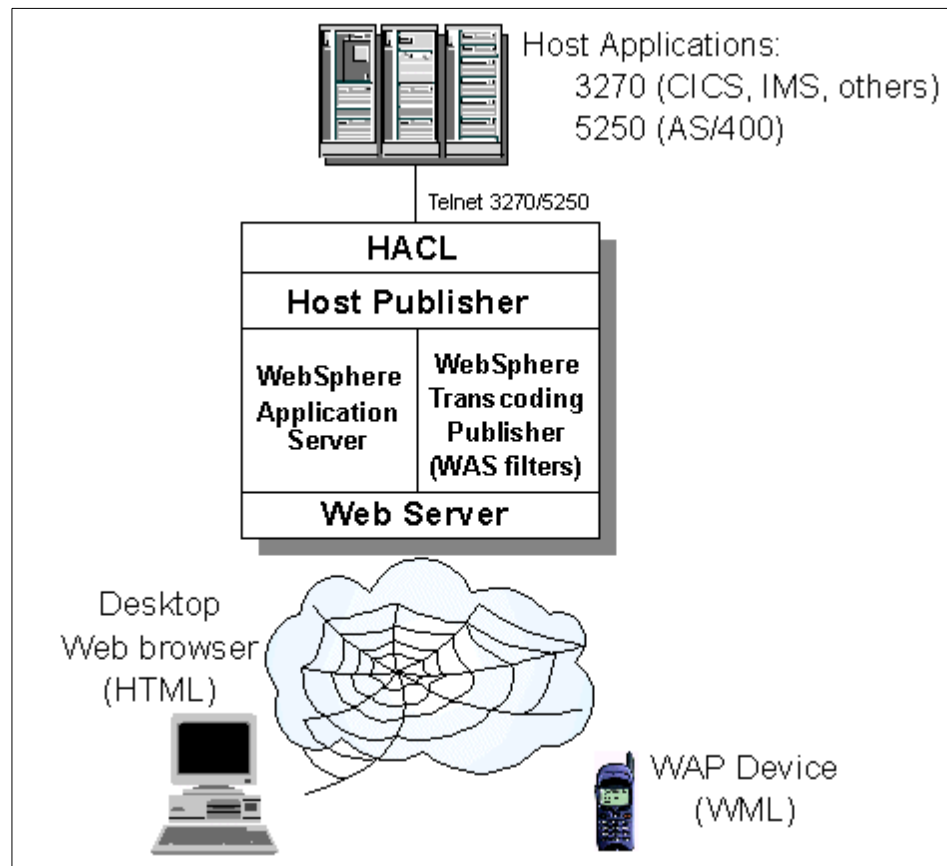


Figure 195. Transcoding Host Publisher applications with WAS filters

11.3.1 Sample Host Publisher application

In this sample scenario, we create a Host Publisher application to run an inquiry CICS transaction as shown in Figure 196 on page 240.

11.3.1.1 Configuration

In order to configure this scenario, we recommend the following steps:

1. If you do not have a Host Publisher application:
 - a. Create the required integration objects using the Host Publisher Studio Host Access.

- b. Create the Host Publisher application (JSPs) using the Host Publisher Studio. Typically you will create a JSP including a form (input page), an execution page to invoke the integration object and an error page.
2. Using the Host Publisher Studio, transfer the application to the Host Publisher Server (runtime).
3. Using the Host Publisher Administration servlet, deploy your application into the production sub-directories.
4. In WAS administration, define a filter for the JSP files.
5. Verify the result using an Internet browser.
6. Install the WAP simulator.
7. Create a WAP application.
8. Run the WAP application.

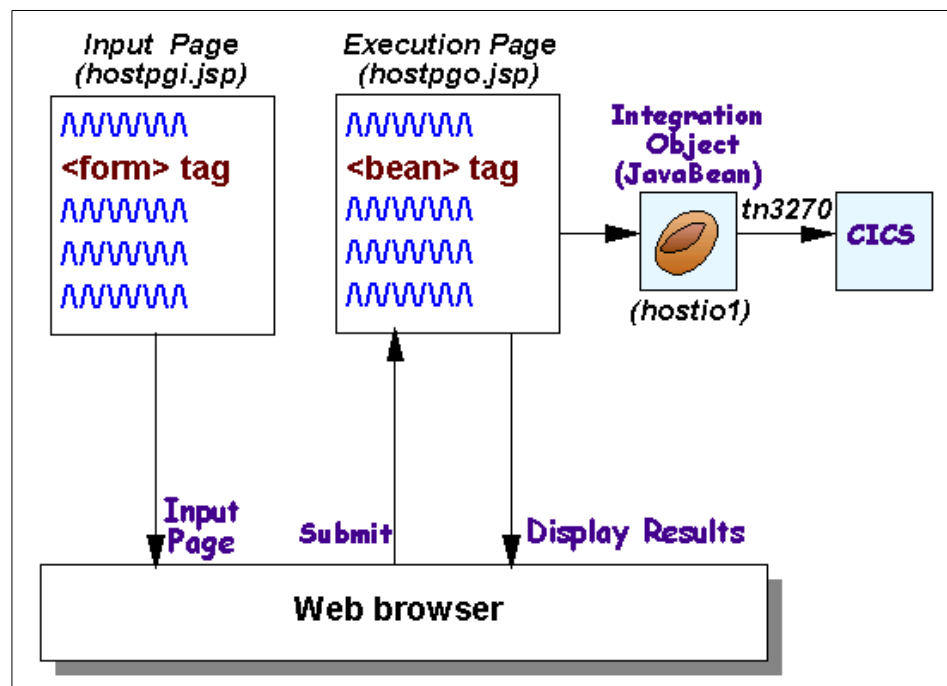


Figure 196. Host Publisher sample application (hostapp1)

11.3.1.2 Creating the host integration object

The Host Publisher integration object (host IO) is a JavaBean and it is created using the Host Access option in the Host Publisher Studio. The IO contains the business logic to access CICS and execute an inquiry transaction based

on input from the client device (account number). The integration object returns the reply from CICS and it is extracted as plain text.

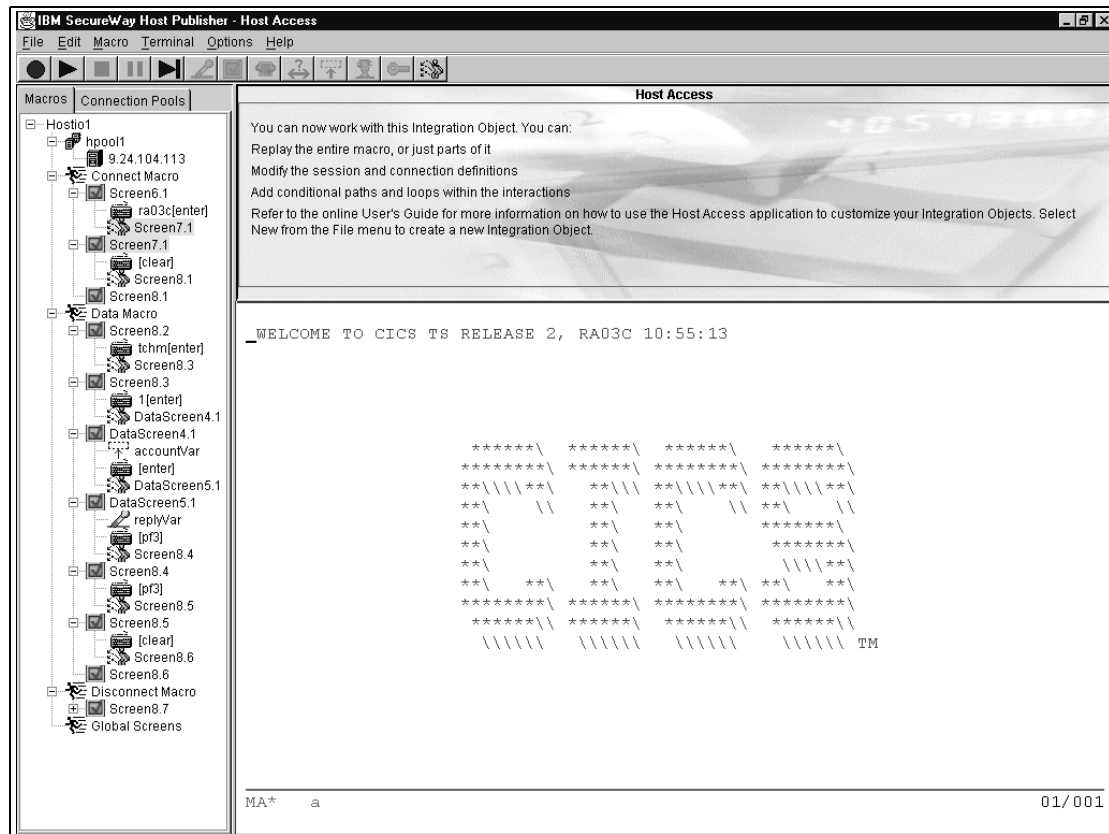


Figure 197. Host Publisher Studio - macros and CICS welcome window

Figure 197 illustrates the Host Access component which is part of the Host Publisher Studio. The CICS connections allow you to create the integration object and the XML files for the following:

- Connection Pool. It contains information about your host connection pool.
- Host connection. It contains information about the host connection.
- Connect macro. This is the login macro to access the host application.
- Data macro. This the actual host transaction.
- Disconnect macro. This is the logoff macro.

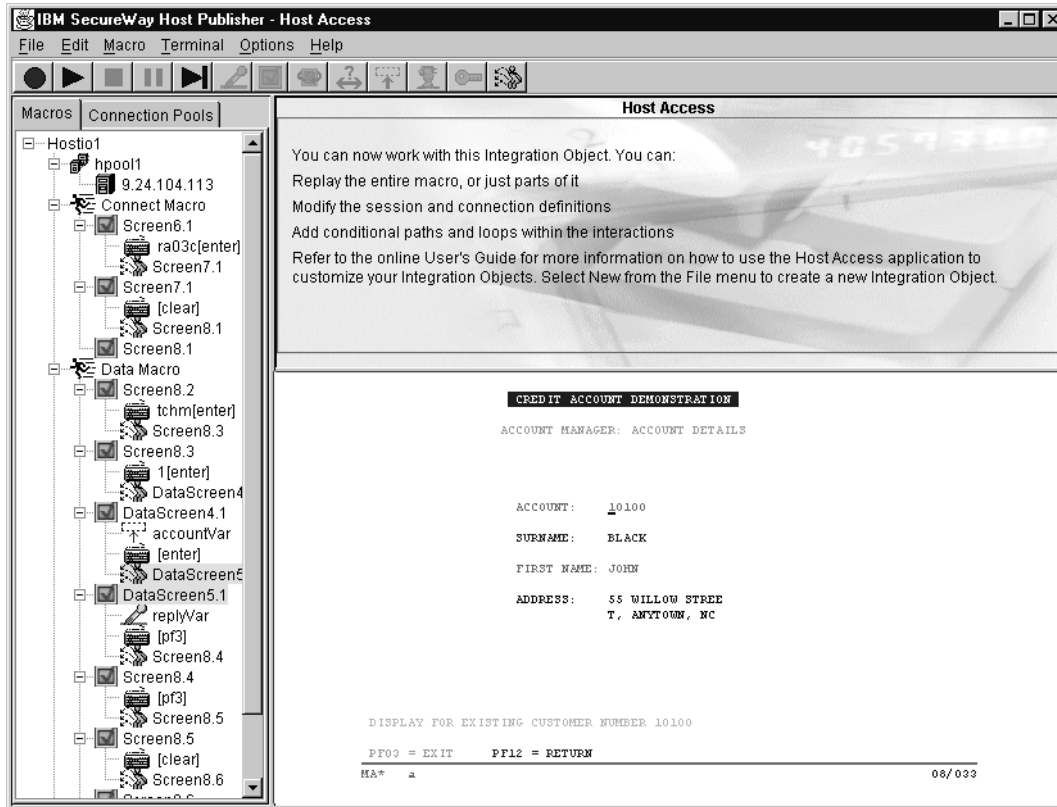


Figure 198. Host Publisher Studio - data macro with CICS results

11.3.1.3 Creating the Host Publisher application (JSPs)

The Host Publisher Studio is used to create the Host Publisher application. It consists of the following JSPs:

- Input page. It contains a form to request input from the client device. The form will post the execute page.
- Execute and display page. It contains the logic to receive the input parameters from the input page (Submit), invoke the host integration object (CICS inquiry transaction) and display the results.
- Error Page.

Figure 199 shows the input page which includes the form requesting input for the account number to be used in the CICS inquiry transaction. Notice that the form in this input page will post the output page `hostpgo.jsp` when the Submit button is used.

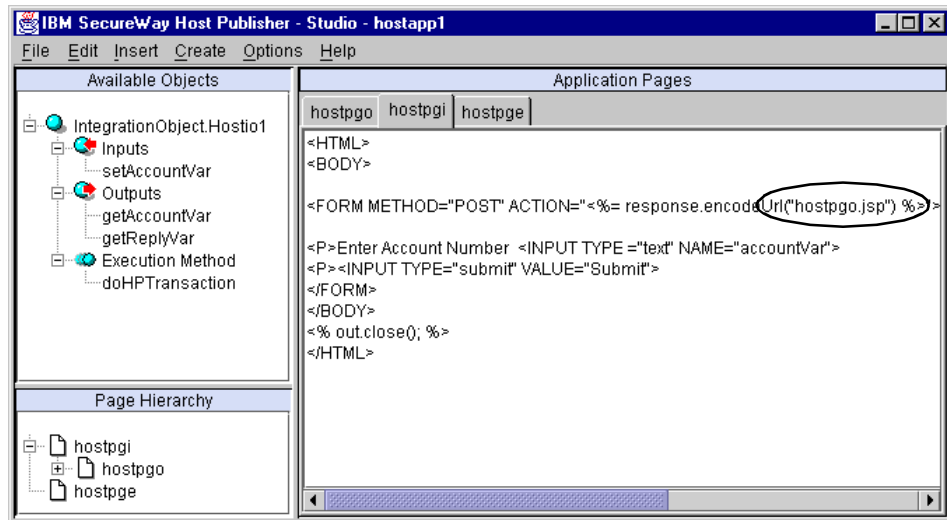


Figure 199. Input page (hostpgi.jsp)

When a user selects the Submit option, the execute page is posted to process the form (see Figure 200).

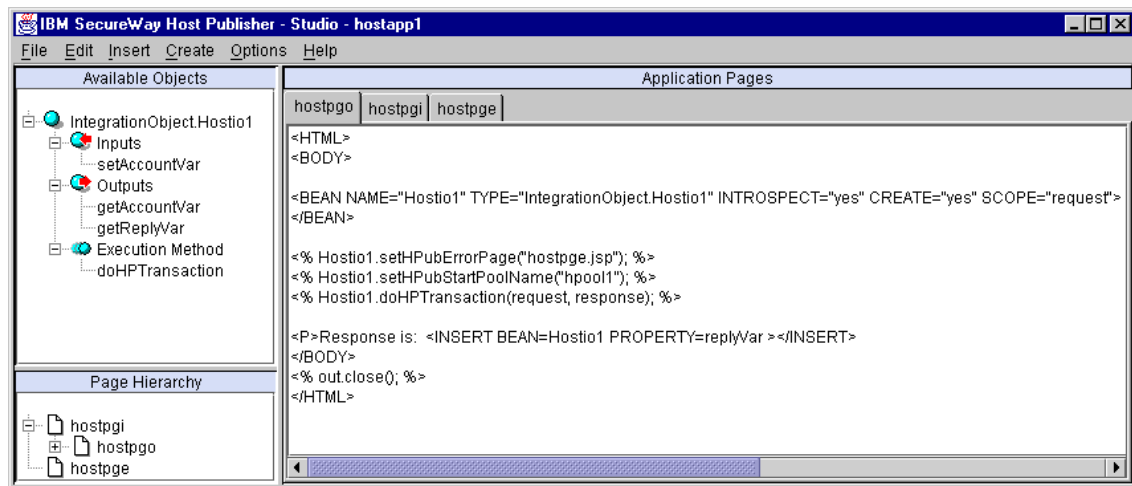


Figure 200. Execution and display page (hostpggo.jsp)

11.3.2 Transfer application to server

Once you have created the integration object (JavaBean) and the application (JSPs), you will transfer the entire application to the server. For this process,

you will use the Transfer to Server option in the Host Publisher Studio as shown in Figure 201.

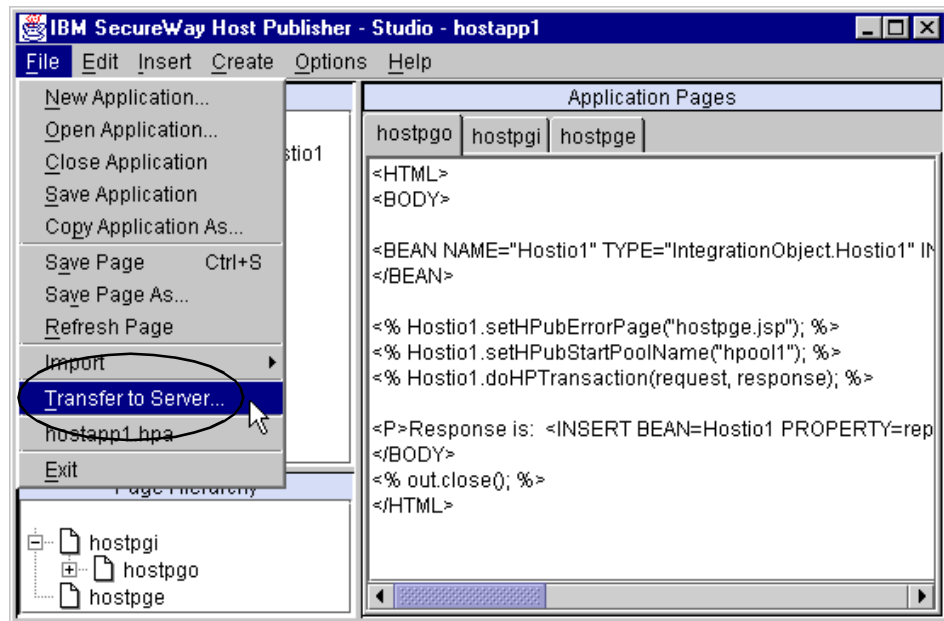


Figure 201. Transfer Host Publisher application to server

11.3.3 Deploy application

After you have transferred your Host Publisher application to the server, the application must be deployed into production. The Host Publisher Administration servlet is used for this function as illustrated in Figure 202 on page 245.

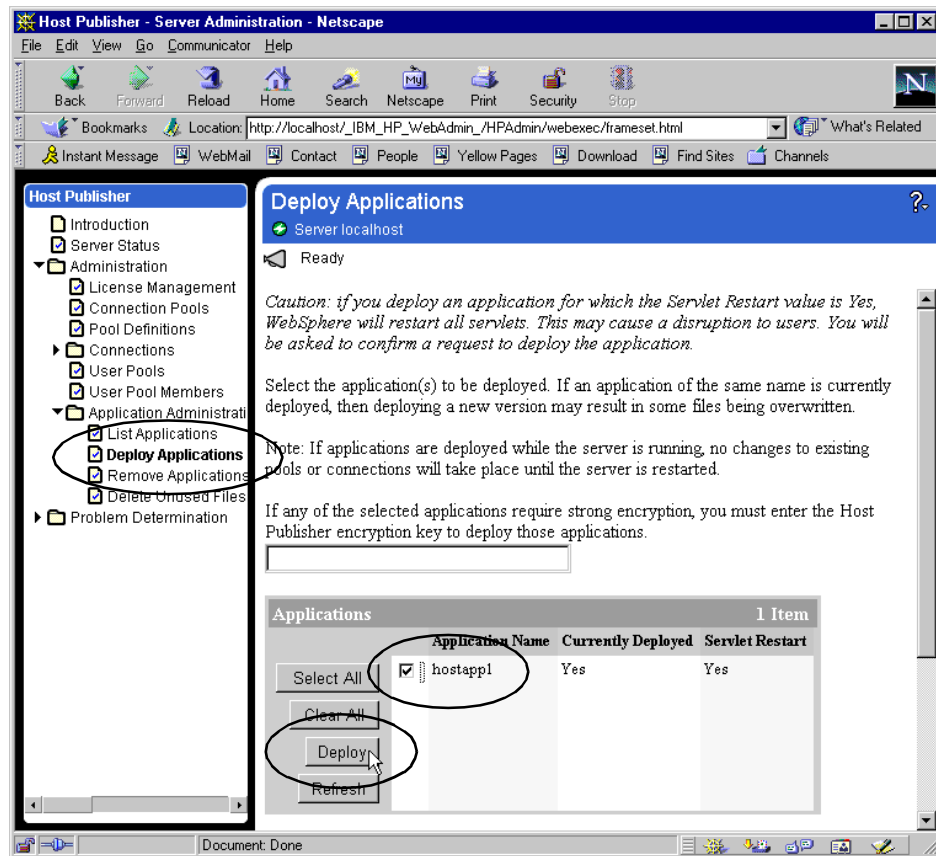


Figure 202. Deploy Host Publisher application (hostappl)

11.3.4 Defining the transcoding filter

Use WebSphere Application Server Administration to enable transcoding filtering by defining the filter for the proper MIME-type your application produces and by redefining the alias for the JSP pages. The latter allows the chaining of the TranscodingPreamble servlet and the JSP pageCompile servlet. The process can be simplified as follows:

1. In Windows NT, invoke WebSphere Administration remotely with the proper URL (`http://<server-name>:9527/`) or from the WAS server desktop. For example:

**Start->Programs->IBM WebSphere->Application Server
V2.0->Administration**

2. In the Filtering pane, enter the MIME-type your application produces and that you want to transcode. In Host Publisher, it is text-html and the WTP servlet (WAS filter) is TranscodingFilter as illustrated in Figure 203.

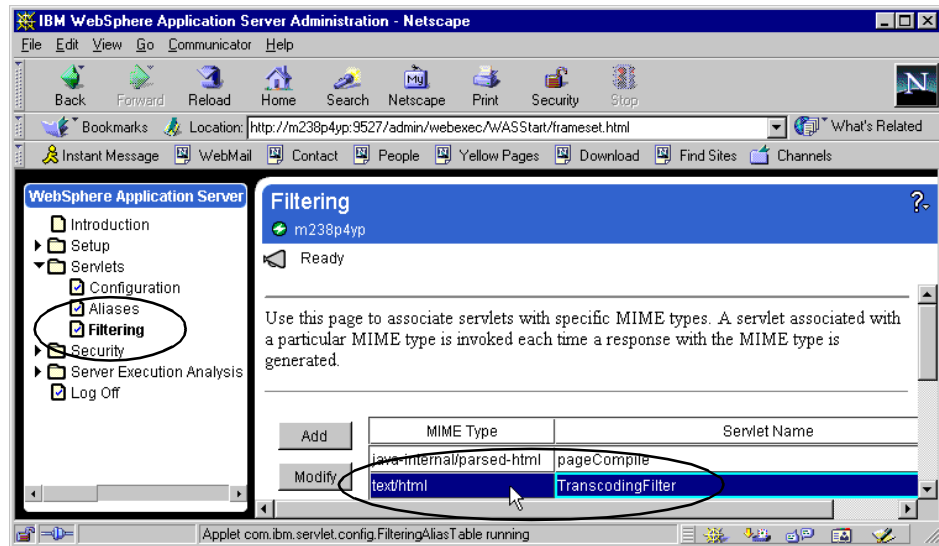


Figure 203. WAS Administration - Filtering

3. In the Aliases pane, configure chaining for the TranscodingPreamble and the JSP pageCompile (see Figure 204 on page 247). As previously mentioned in this chapter, all JSP content will be transcoded.

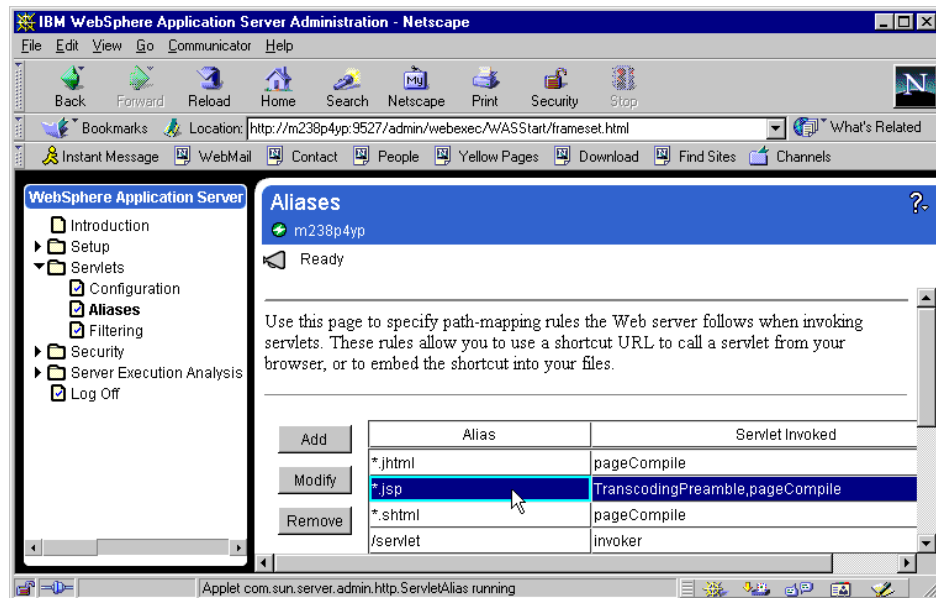


Figure 204. WAS Administration - chaining configuration

In some cases, after entering your configuration you will need to restart the WebSphere Application Server to have your changes take effect. For example, after switching to and from servlet mode you will need to shutdown WAS, make any changes and restart.

11.3.5 Invoking transcoded content from a desktop browser

In this section we show how the Host Publisher application is invoked using a Netscape browser and using the proper URL as follows:

`http://<computer name>/HostPublisher/hostapp1/hostpg1.jsp`

where:

<computer name>

is the computer name of the Host Publisher server.

HostPublisher

is the default Host Publisher alias in the Web server.

hostapp1

is the Host Publisher application name.

hostpg1.jsp

is the JSP input page in this sample application.

The input page `hostpgi.jsp` displays the form and request and account number to be passed to the integration object that will execute the CICS inquiry transaction. For example, we enter the account number 10100 as illustrated in Figure 205 on page 248.

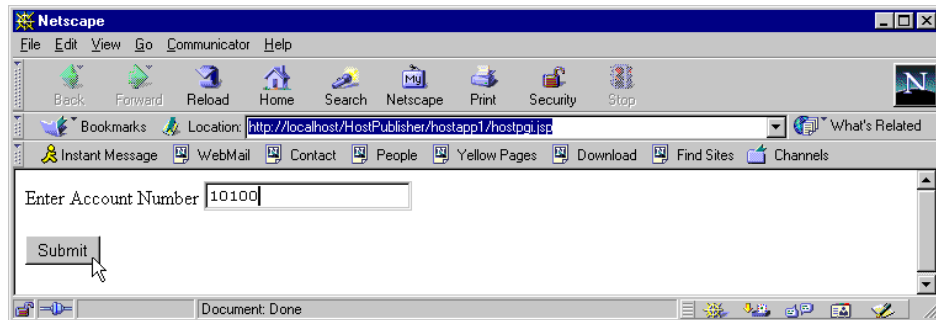


Figure 205. Host Publisher sample application input page form

Figure 206 shows the HTML code received at the Web browser for the Host Publisher input page in this sample application (`hostpgi.jsp`).



Figure 206. HTML page created from the JSP input page

When you submit the form in the input page the following steps take place:

1. The execution page `hostpg0.jsp` is posted.
2. The `TranscodingPreamble` servlet detects the client type (user agent) and passes this to the `TranscodingFilter` servlet.
3. The Host Publisher execution page invokes the integration object (JavaBean).

4. The integration object sends the CICS transaction using the Telnet 3270 connection (via HACL).
5. CICS processes the transaction and sends the response via the 3270 session.
6. The Host Publisher execution page gets the response from CICS (as text in this sample scenario).
7. The HTML page containing the result is dynamically built.

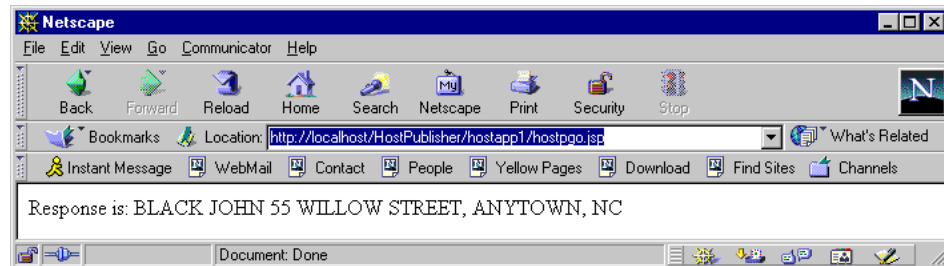


Figure 207. CICS response rendered as plain text

In this sample application, although Transcoding Publisher was used as WAS filters, there was no change in the HTML since the Web browser supports this markup language. It is included here to show the mechanism of the Web integrated 3270 CICS transaction.

Figure 208 shows the received and dynamically created HTML page including the response from the CICS inquiry transaction.

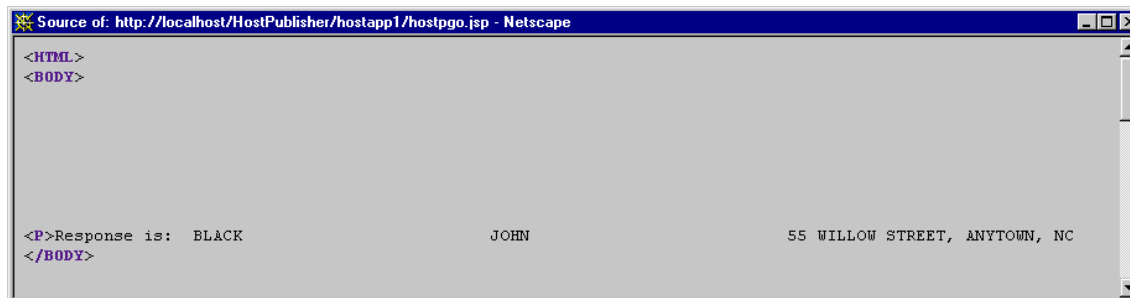


Figure 208. HTML source for the dynamically created CICS transaction page

11.3.6 Transcoding HTML to WML from JavaServer Pages

In this section we show you the same Host Publisher application described in 11.3.1, "Sample Host Publisher application" on page 239 but this time it will

be accessed from a WAP device supporting WML. We use the Nokia WAP toolkit for this purpose.

For information about how you download a trial version, install and configure the Nokia WAP toolkit see <http://www.forum.nokia.com>.

11.3.6.1 Creating an Initial WAP application

Before you actually invoke the Host Publisher application, you will need to create a WAP application (WML) that will be used as a menu in the WAP device. For example, Figure 209 shows a WML file containing selections to access a demo program (TxDemoWAP.jsp) and the Host Publisher application (hostpgi.jsp).

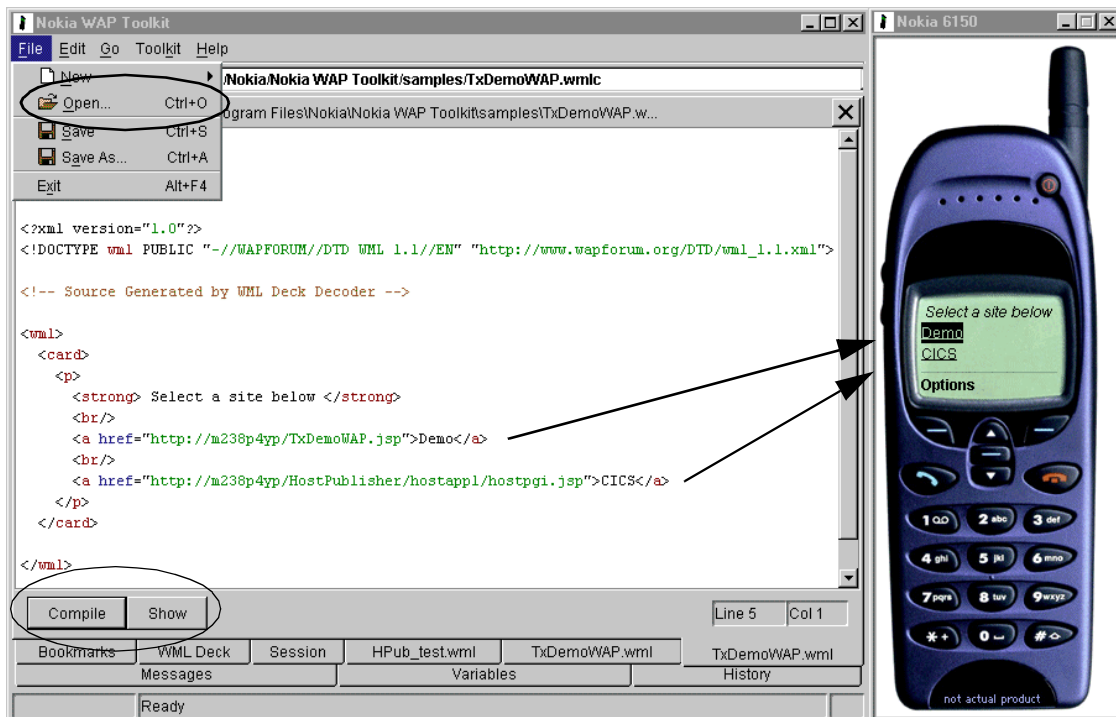


Figure 209. WML initial WAP application (TxDemoWAP.wml)

11.3.6.2 Running the initial WAP application

In the Nokia WAP Toolkit, follow these steps to run the demo and Host Publisher applications:

1. Start the Nokia WAP toolkit (see Figure 211 on page 252).

2. Open, compile and show the TxDemoWAP.wml sample shown in Figure 209.
3. Select the demo program to invoke the TxDemo.jsp and obtain the transcoded (HTML to WML) results. The JSP used in this demo is shown in Figure 210.

```
<HTML>
<HEAD>
<TITLE>Hello World - JSP</TITLE>
</HEAD>

<BODY>
<H1>This is a test using Transcoding Publisher and WAP</H1>
</BODY>
</HTML>
```

Figure 210. A simple JSP to exercise HTML to WML transcoding (TxDemo.jsp)



Figure 211. Executing a simple JSP to transcode HTML to WML

Click on **WML Deck** to display the transcoded WML code for the simple JSP we are using. Figure 212 on page 253 shows the transcoded WML.

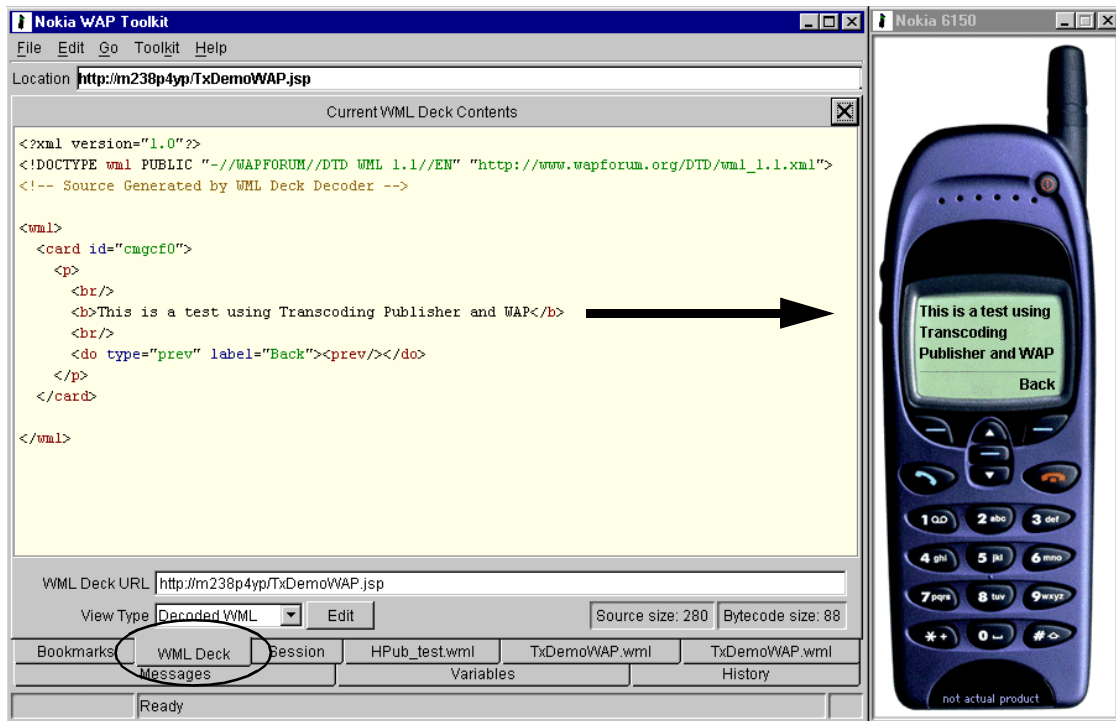


Figure 212. Transcoded WML for the demo JSP

Now that WTP is transcoding the JSP contents, we proceed to invoke the CICS inquiry transaction using the Host Publisher application we created in 11.3.1.3, "Creating the Host Publisher application (JSPs)" on page 242.

11.3.6.3 Transcoding Host Publisher content (HTML to WML)

Follow these steps to run the CICS transaction:

1. From the initial WAP page, select the CICS option (see Figure 213 on page 254).
2. The input page hostpgi.jsp is invoked and its output is transcoded. Figure 214 on page 255 illustrates the transcoded WML code for this JSP input page containing a form requesting an account number. The account number will be used as input to the CICS transaction. Next, we enter a valid account, for example 10100.
3. Select the **Submit** option as illustrated in Figure 213 on page 254.
4. The request is sent to post the Host Publisher execution page (hostpgi.jsp) and the following process takes place:

- a. WTP TranscodingPreamble detects the client device (user-agent).
- b. The execution JSP invokes the host integration object.
- c. The integration object uses HACL to send the CICS transaction via the Telnet 3270 protocol.
- d. CICS receives the transaction, it processes the request for the entered account number and accesses the database to obtain the record information for this account.
- e. CICS sends the response using the 3270 data stream.
- f. Host Publisher extracts the response to be dynamically inserted into the execution (output) page. HTML code is produced.
- g. WTP, configured as a WAS filter, transcodes the generated HTML code (containing the CICS response) into WML code to be sent to the client device.



Figure 213. Executing a CICS transaction from a WAP device

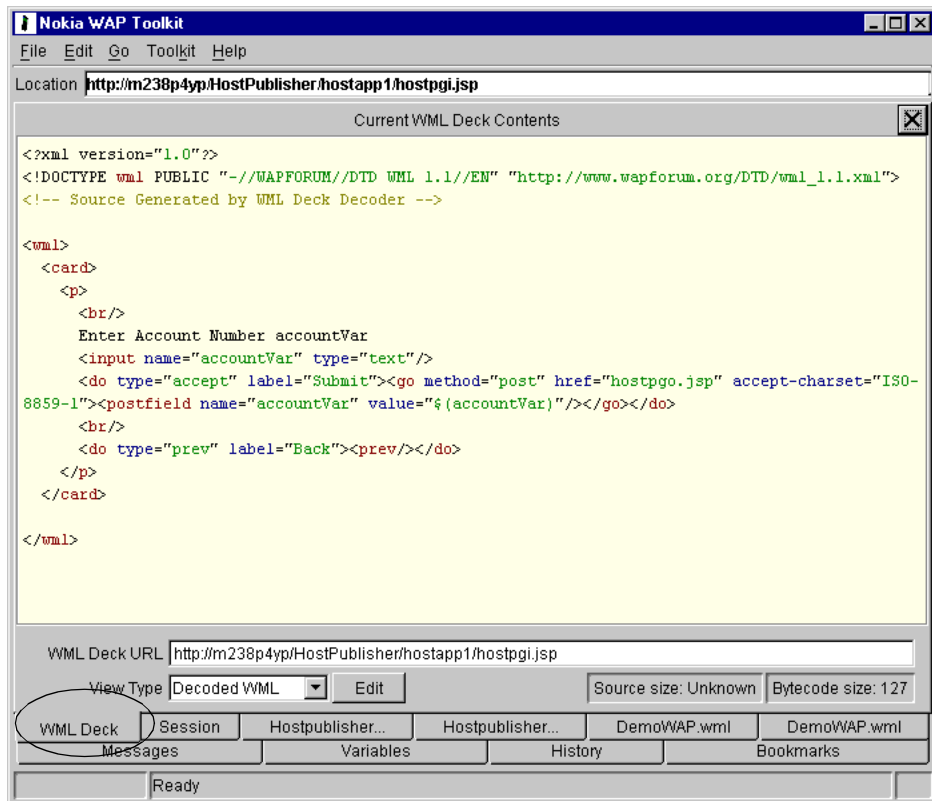


Figure 214. Transcoded WML code for the input page (hostpgi.jsp)

The initial Host Publisher application input page (hostpgi.jsp) includes a form to request input using a form. The page was previously created using the Host Publisher Studio as described in 11.3.1.3, “Creating the Host Publisher application (JSPs)” on page 242.

The HTML code is passed to the WTP filter (in the WebSphere Application Server) along with the client information (user agent) to be translated into WML code. Figure 214 shows the transcoded WML code received by the client device. The page displays a form requesting an account number and it will post the execution (and output) page (hostpggo.jsp) when the Submit button is clicked.

In this scenario the Nokia WAP toolkit is the client device supporting WML. Click the **WML Deck** option to display the transcoded WML page.

When the Submit button is pressed, the request flows to the WAS server to post the execution page (hostpgo.jsp). After the host integration object is invoked to execute the CICS transaction, HTML code is dynamically created to include the CICS reply. Next, the WTP filter is called to translate the produced HTML code into WML.

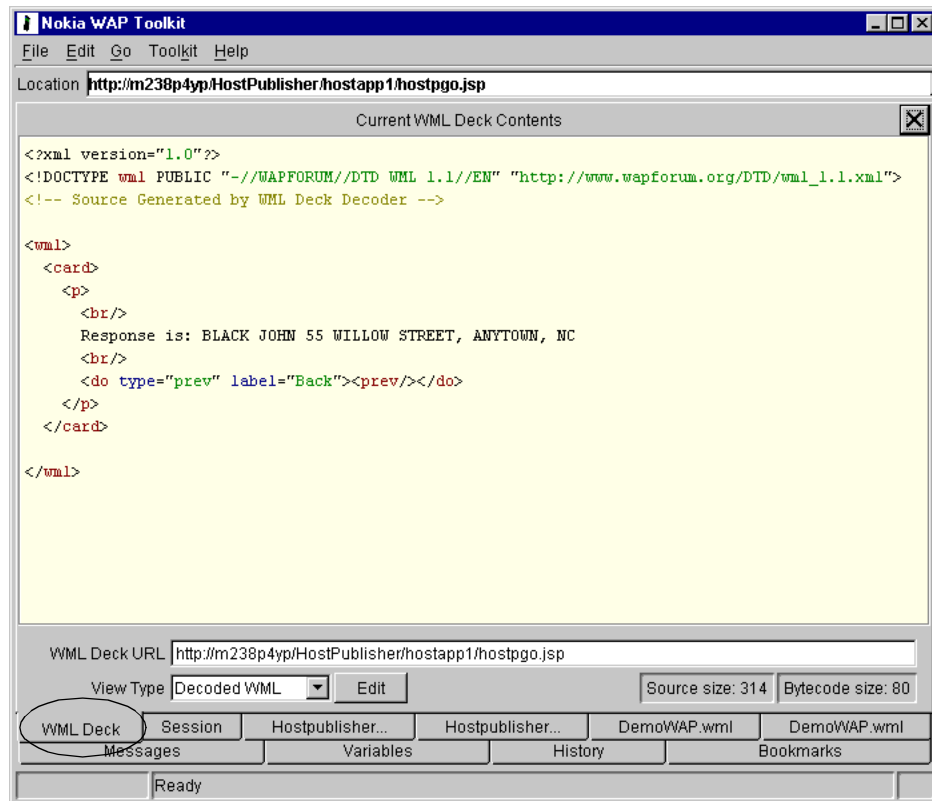


Figure 215. Transcoded WML code for the output page (hostpgo.jsp)

Again, click **WML Deck** to display the received WML code. It is shown in Figure 215.

11.4 Scenario: transcoding using WTP as a network proxy

You can also transcode Host Publisher applications using IBM WebSphere Transcoding Publisher as a proxy. In this section we show you a sample configuration and mention a few hints you may need to be aware of.

To configure this scenario, you need to have the proxy and Host Publisher scenarios already configured and running. We use the same proxy that was used for the Chapter 10, “Transcoding wireless applications” on page 207 and the same application developed for 11.3, “Scenario: transcoding with WAS filters” on page 238.

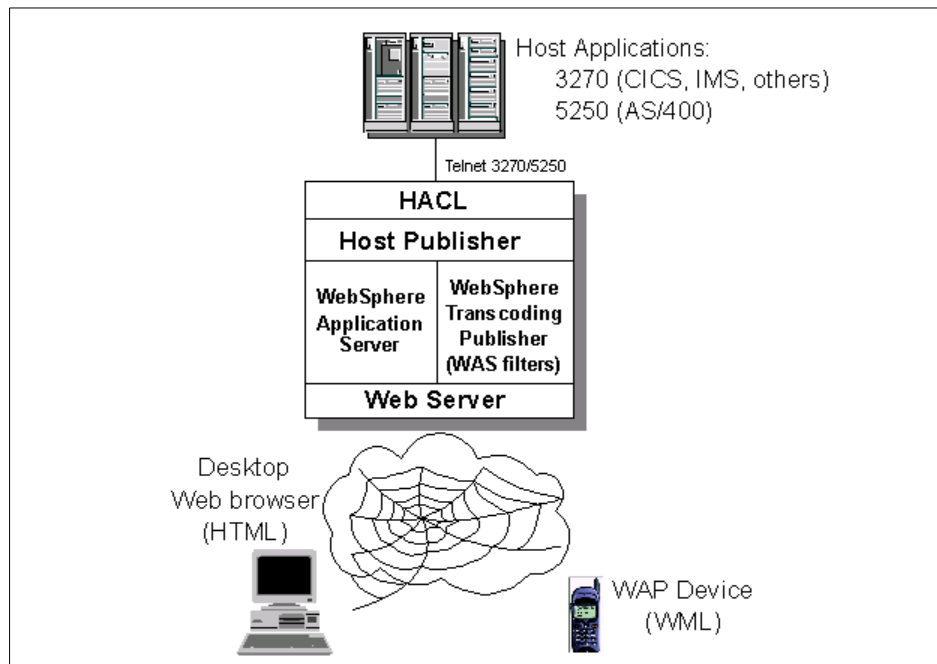


Figure 216. Scenario: transcoding Host Publisher with WTP as a proxy

The only reason we are using WebSphere Application Server is because Host Publisher requires it.

11.4.1 Configuration

We recommend the following steps to transcode Host Publisher applications with WTP running as a proxy:

1. If WTP has been installed to run as a WAS filter, leave WTP servlet mode so that WebSphere Application Server (WAS) is not configured to use WebSphere Transcoding Publisher (WTP) servlets as filters. To accomplish this, go to the WebSphere Application Server Administration Utility and click **Filtering** under the Servlets option. Figure 217 on page 258 indicates that WTP is not configured as a WAS filter.

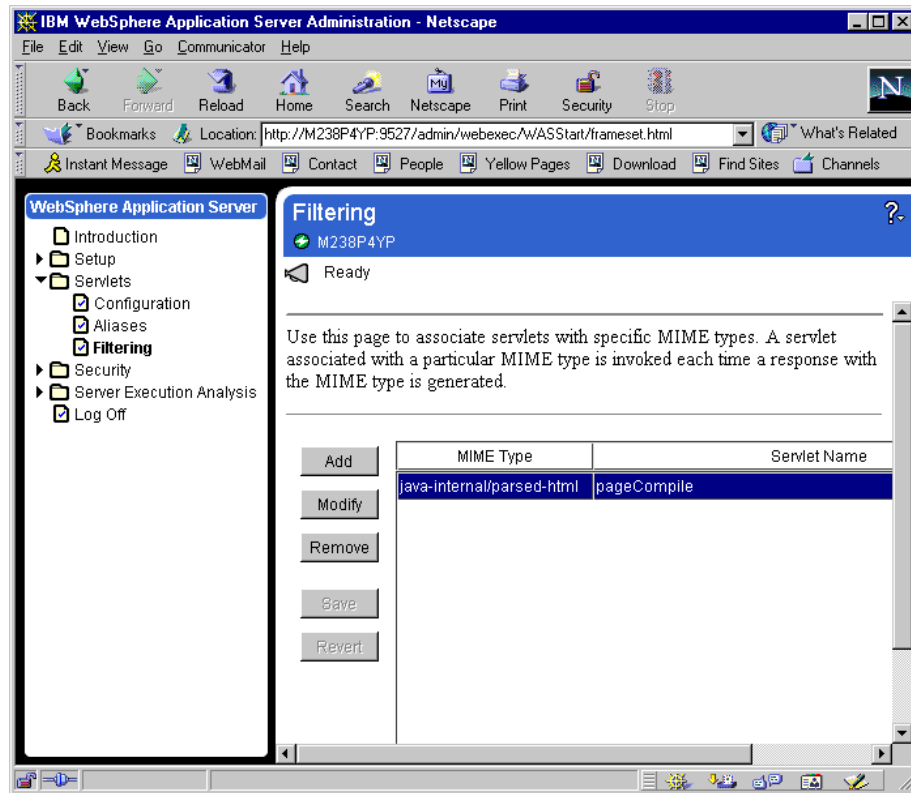


Figure 217. WebSphere Application Server Administration utility

2. Make sure that you have installed and configured WTP to run as a proxy server. See Chapter 5, “Running as a stand-alone network proxy” on page 63 for details.
3. In our scenario, using the Nokia toolkit, you will also need to configure the HTTP proxy settings on the WAP emulator, if it is not there already. Open the WAP Toolkit Emulator and select **Preferences** under the Toolkit menu. Add the address for your WTP proxy and the port number associated with the Wireless network profile (8089 by default) as shown in Figure 218 on page 259.



Figure 218. WAP emulator - proxy configuration

Now, use the same application that you created for the Host Publisher scenario using WTP as a WAS filter and repeat the same procedure to navigate on the WAP emulator as described in 11.3.6.3, “Transcoding Host Publisher content (HTML to WML)” on page 253.

In most cases, you will have similar results and this is because WTP is using the same transcoders but in different ways. However, in some cases you may have slightly different results due to the following:

1. In most cases, the network profile is used when running WTP as a proxy. However, it is never used when WTP is configured to run as a WAS filter.
2. In this release, WML fragmentation is not used if you have configured WTP as a WAS filter. When using WTP as a proxy, WML fragmentation will break the result into as many pages as needed so that the result can fit into the available memory. WML fragmentation will be enhanced to run in servlet mode (WAS filter) in a future release.

Chapter 12. Administration

The primary tool for administering Transcoding Publisher is the Administration Console. This chapter describes the console and its functions, and provides instructions for most of the common administrative tasks.

12.1 Supported tasks

The IBM WebSphere Transcoding Publisher Version 1.1 (WTP) product provides an Administration Console. It is the graphical interface for your administrative tasks such as:

- Working with resources
 - a. Enabling and disabling preference profiles, transcoders, and stylesheets
 - b. Modifying preference profiles and stylesheet selectors
 - c. Adding new preference profiles, transcoders, and stylesheets
 - d. Organizing stylesheets into folders
- Working with settings
 - a. Modifying firewall settings
 - b. Modifying cache settings
 - c. Modifying port settings for network types
- Refreshing the server
- Working with logging and tracing

12.2 Using the Administration Console

The IBM WebSphere Transcoding Publisher Administration Console provides the following functions:

- A tree view of your resources (preference profiles, transcoders, and stylesheets)
- The ability to edit individual resources
- A menu structure to provide access to other tasks
- Help for each panel, as well as a list of How do I ... topics

The WTP Administration Console is started from the desktop. For example, in a Windows NT, select:

Start->Programs->IBM Transcoding Publisher-> Administration Console.

In AIX, Sun Solaris, and Linux, enter *AdminConsole.sh* at the command line.

12.2.1 The main panel

The main panel of the Administration Console has two panes. The left pane shows a tree view of your resources.

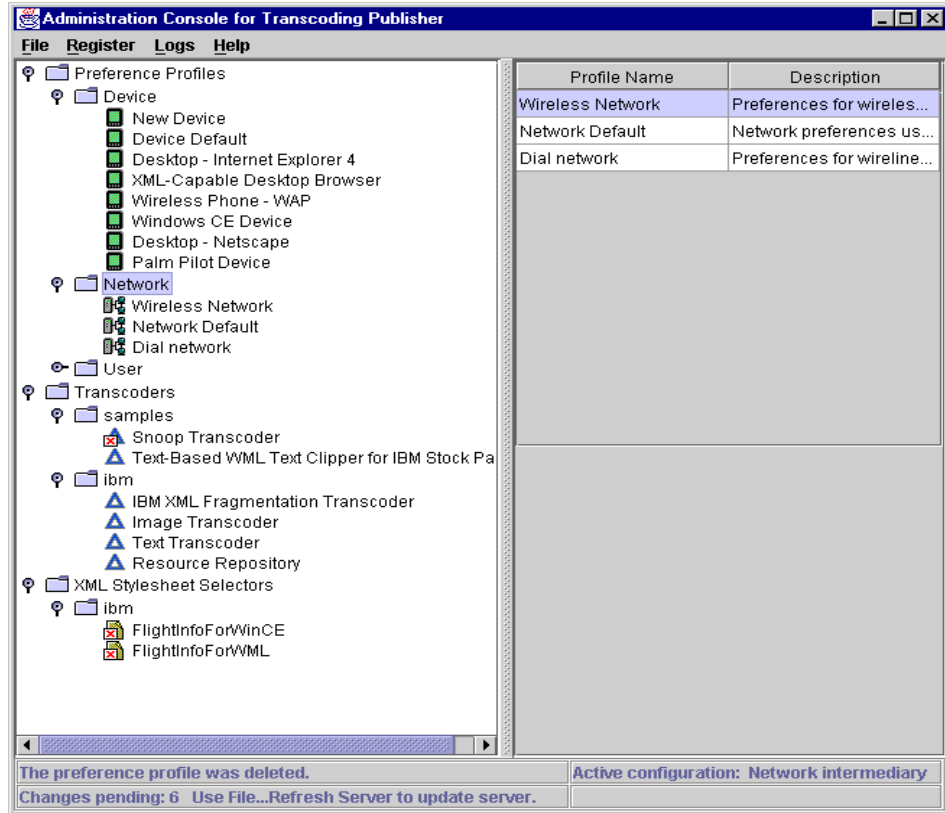


Figure 219. Administration Console main panel

A turner icon appears to the left of each folder name. Figure 220 shows these symbols and their meaning.

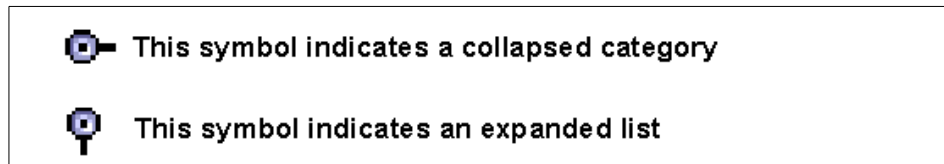


Figure 220. Administration Console - collapsed and expanded categories

Each transcoder, preference profile, and stylesheet resource is represented by a colored icon.

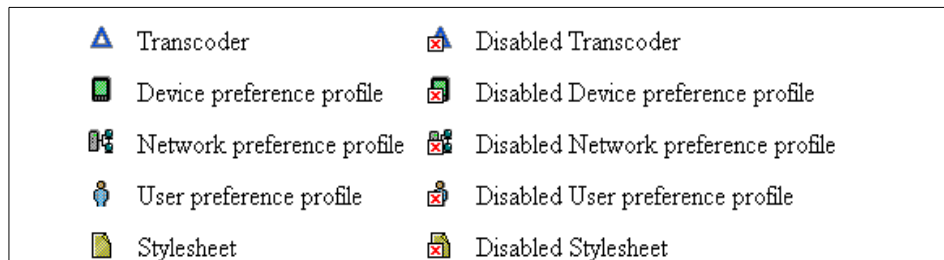


Figure 221. Icon representation of resources

When you select a resource in the left pane, its properties will be displayed in the right pane, where you can view and modify them. When you select a folder in the left pane, the names and descriptions of the resources in that folder are displayed as a list in the right pane. You can then double-click on a list item to work with it.

The status area at the bottom of the Administration Console displays the active configuration you are using, either WebSphere filter, network intermediary, or caching network intermediary. You will also see the latest action taken and a message field (see bottom of Figure 219 on page 262).

12.2.2 Working with resources

IBM WebSphere Transcoding Publisher uses three kinds of resources:

- Preference profiles
- Stylesheets
- Transcoders

You can use the Administration Console to work with your resources in several ways.

- Enabling and disabling preference profiles, transcoders, and stylesheets:

If you have a preference profile, transcoder, or stylesheet that you don't want IBM WebSphere Transcoding Publisher to use, you can disable it. It will still be registered and available for IBM WebSphere Transcoding Publisher to use whenever you want to resume using it.

To disable a resource, select it in the tree view to display its properties in the right pane. Click the **Enabled** checkbox to deselect it. After you click **Save and refresh the server**, this resource will not be used by IBM WebSphere Transcoding Publisher.

To enable a resource that you have disabled, or to begin using a resource if you did not enable it when you added it to IBM WebSphere Transcoding Publisher, select the resource in the tree view to display its properties in the right pane. Click the **Enabled** checkbox to select it. After you click **Save and refresh the server**, this resource will be used by IBM WebSphere Transcoding Publisher.

When you disable a resource, its icon is changed to contain a red X in a white box, as shown in Figure 221 on page 263. This makes it easy to identify disabled resources.

- Modifying preference profiles and stylesheet selectors

You might want to modify an existing preference profile to change how Transcoding Publisher handles documents requested by a particular device or network. For example, you might decide you want to change image processing to favor download speed over image quality more than the profile currently specifies. To modify a profile, select it in the tree view to display its properties in the right pane. You can modify any of the displayed properties. Click **Advanced** if you want to add criteria to select a stylesheet to be used when XML documents are processed using this profile.

You may want to modify the conditions under which a stylesheet is selected to process XML documents. To modify a stylesheet selector, select it in the tree view to display its properties in the right pane. You can modify any of the displayed properties, including the output content type and the input or output document type definitions (DTDs). Click **Advanced** for more options.

From the Advanced panel you can specify two types of additional criteria for selecting a stylesheet:

- Use the Criteria matching HTTP header box to enter a formula that must be true for this stylesheet to be used.

- Use the Criteria matching preferences box to specify criteria that will cause this XML stylesheet to be associated with a preference profile.
- Adding new preference profiles, transcoders, and stylesheets

You can extend IBM WebSphere Transcoding Publisher's functionality by adding new stylesheets, transcoders, or preference profiles. You add new resources to IBM WebSphere Transcoding Publisher by registering them. After you register a new resource, you will see it in the tree view of the Administration Console. After you have registered all your new resources, you must refresh the server so that IBM WebSphere Transcoding Publisher can use them. Select **Register** and then the appropriate resource.
- Deleting resources

If you wish to completely delete a resource that you have added to IBM WebSphere Transcoding Publisher, highlight the resource in the tree view and select **File->Delete**. You cannot delete the three base folders (Stylesheets, Transcoders, and Preference Profiles) or resources such as transcoders provided with IBM WebSphere Transcoding Publisher.
- Organizing stylesheet selectors into folders

You may find it useful to organize your XML stylesheet selectors into folders. This arrangement of stylesheet selectors and folders is solely for your convenience in finding and working with stylesheets. Moving a stylesheet selector to a different folder does not affect its file location or selection criteria.

The two operations you can use to organize your stylesheet selectors are creating new folders and moving a stylesheet from one location to another. A new folder must be created within an existing folder, which can be the base folder of Stylesheet Selectors. To create a new folder, select the folder under which you want to create the new folder, then select **File->New stylesheet** folder. Enter a name for the new folder and click **OK**. The new folder will appear in the tree.

After you have set up your folders, you can move stylesheet selectors from one folder to another. To do this, select the stylesheet selector, then select **File->Move XML Stylesheet**. In the Move Resource dialog box, select the location in the tree to which you want to move the stylesheet selector.

You can delete stylesheet selectors or folders by selecting the item in the tree and then selecting **File->Delete**.

12.2.3 Working with settings

Depending on your active configuration, you may be able to modify your firewall settings, cache settings, proxy port settings, and server setup.

Table 4. WTP configuration and options

Configuration	Configurable options
WebSphere filter	server setup
Network proxy	firewall settings, proxy port settings, server setup
Caching network proxy	firewall settings, cache settings, proxy port settings, server setup

If you want to change your active configuration, such as adding or deleting a cache server or switching from a WebSphere filter to a network proxy, you must use the Server Setup wizard. Select **File-> Server Setup**.

Modifying firewall settings

If you use a firewall in your network, you must supply the address and port number on which IBM WebSphere Transcoding Publisher will communicate with it. You can specify the address in dotted decimal format or as a hostname. If you enter a hostname, IBM WebSphere Transcoding Publisher will attempt to validate the hostname through your Domain Name Server (DNS).

Select **File->Settings->Firewall** to use the Firewall Settings panel to modify the address and port number. On this panel you can also list addresses within your network with which IBM WebSphere Transcoding Publisher can communicate directly, without going through the firewall server.

Modifying cache settings

Cache settings will be available only if you have configured IBM WebSphere Transcoding Publisher as a network proxy that uses a cache. Select **File->Settings->Cache** to view the Cache Settings panel.

The Cache Settings panel contains a list of defined MEG groups. If you want to change the order in which MEG groups are executed, use this list to move one or more MEG groups up or down within the list.

Modifying port settings for network types

Use the Proxy Port Settings panel to specify port numbers for each network type for which you have a preference profile. To open this panel, select **File->Settings->Proxy Port**.

12.2.4 Refreshing the server

When you make changes through the Administration Console, they will take effect only after you refresh the transcoding server. The server must be refreshed when you make any of these changes:

- Enable or disable a resource.
- Modify a preference profile or stylesheet selector.
- Register a new resource.
- Delete a resource.
- Modify firewall, cache, or proxy port settings.
- Change tracing or message logging options.
- Enable or disable tracing or message logging.

You do not need to refresh the server when you reorganize stylesheet selectors by creating new folders or moving stylesheet selectors or folders.

When you make changes that require that the server be refreshed, IBM WebSphere Transcoding Publisher will display a message at the bottom of the Administration Console that says XX changes have been made. Select **File->Refresh Server** to make these changes take effect., where XX is the number of changes you have made since the server was last refreshed.

When you refresh the server, it must reread property files that store information about resources and settings. During this process, transcoding of documents can be delayed. If you are making several changes, do not refresh the server after each change; instead, make all your changes and then refresh the server to implement those changes.

To refresh the server, select **File->Refresh server**. After you do this, you may want to review the message log to verify that the changes took place.

Note: When you make a configuration change that requires refreshing the transcoding server, a message will be displayed in the status area on the Administration Console. If you close the Administration Console while this message is displayed, the server will be refreshed automatically to pick up the changes that you made.

12.2.5 Working with logging and tracing

Message logging and tracing are tools provided as part of IBM WebSphere Transcoding Publisher to help you recognize and debug problems. Logging

and tracing are described in more detail in Chapter 13, “Problem determination” on page 287.

12.2.6 Starting and stopping Transcoding Publisher

IBM WebSphere Transcoding Publisher consists of two separate processes: the Administration Console, which is your interface to configure and administer IBM WebSphere Transcoding Publisher, and the backbone, which does the work of transcoding documents.

If you have configured IBM WebSphere Transcoding Publisher as a WebSphere filter, WebSphere will control the starting and stopping of IBM WebSphere Transcoding Publisher.

If you have configured IBM WebSphere Transcoding Publisher as a network proxy, the method for starting and stopping depends on the operating system. To stop or start IBM WebSphere Transcoding Publisher on Windows NT or Windows 2000, use the Services utility from the Control Panel. On AIX, Sun Solaris, or Linux, use the TransPub script to start, stop, or restart IBM WebSphere Transcoding Publisher.

12.2.6.1 Help

From the Administration Console we can activate:

- Help
- Administrator's Guide
- Contents
- Product Support Site
- Readme
- About

12.3 Creating preference profiles

Preference profiles are the most important definitions we make when we use IBM WebSphere Transcoding Publisher. In these profiles we define what elements are transcoded, and how, for the specific device or network. You should also be aware that in this release, the user profile is only a default user profile.

For information about how to create network and device profiles see 9.3, “Creating preference profiles” on page 174.

Note

Creating a new preference profile is a WTP toolkit procedure. The toolkit is used to create a new preference profile. Once it is created, the WTP Administration Console is used to register the new preference profile.

12.4 Registering preference profiles

New preference profiles must be registered using the Administration Console. In this section we show you how you register network and device profiles.

12.4.1 Registering network preference profile

To register the network profile, start the Administration Console and select:

Register --> Preference Profile

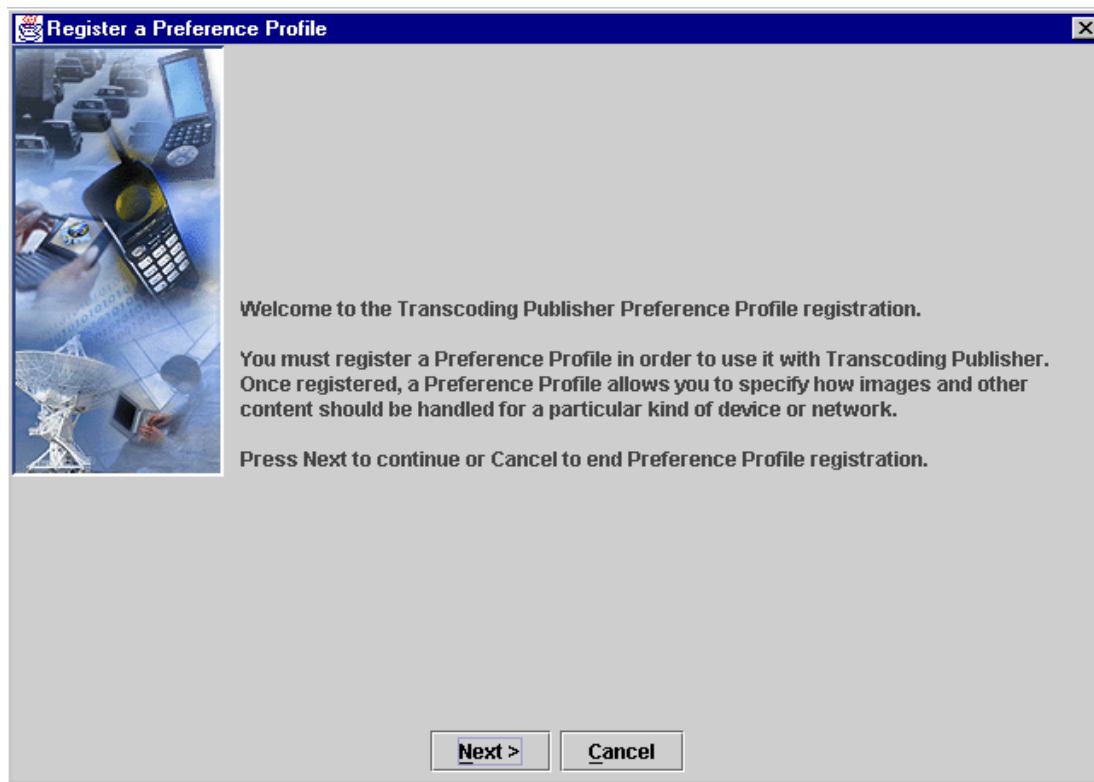


Figure 222. Welcome window to register a profile

Click **Next** and **Browse** to locate the right profile. Our example profile is `newnetwork.prop` in the directory `IBMTrans\etc\preferences\network`.



Figure 223. Selecting the profile to be registered

Click **Next** to define which profile you want to register. Select the **Network** radio button.

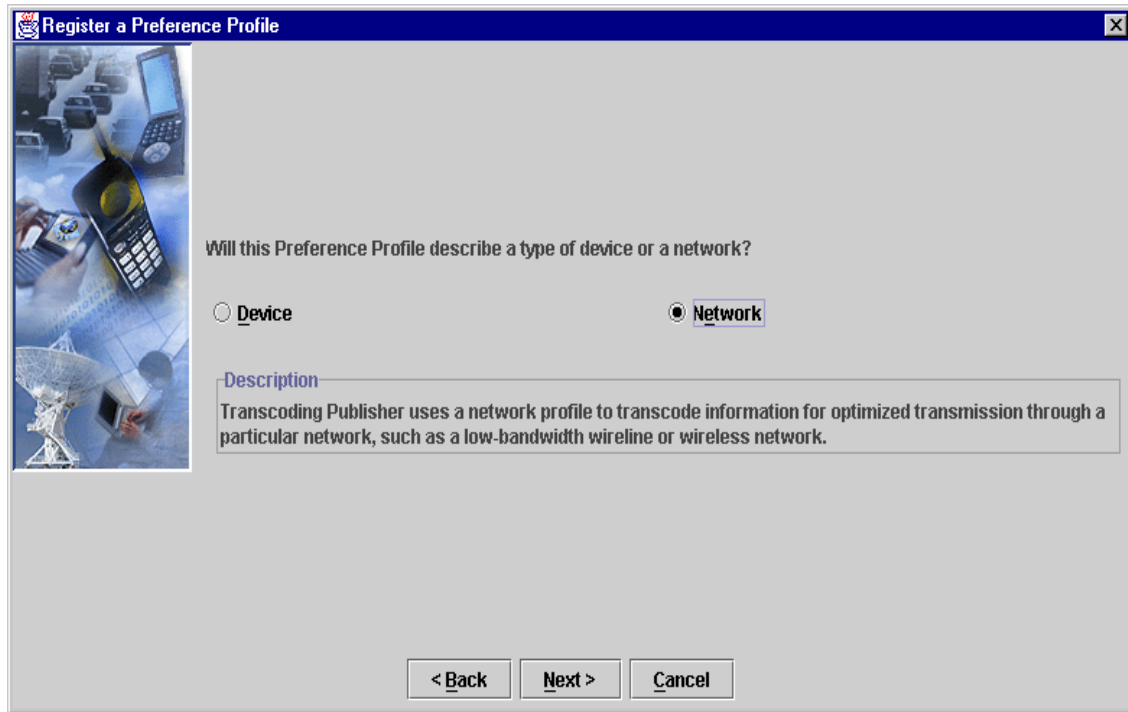


Figure 224. Profile type selection

Click **Next**.

We got a window saying that the profile was already registered. Actually it was not so we had to click Yes. If you select the option No, it will stop the registration process.

We only saw this window because we selected the default directory to store the profile while creating it. If you have created it in any other folder, this message will not appear.



Figure 225. Duplicate profile message

Select **Yes** and then you will have to associate the network profile with a TCP/IP port number.

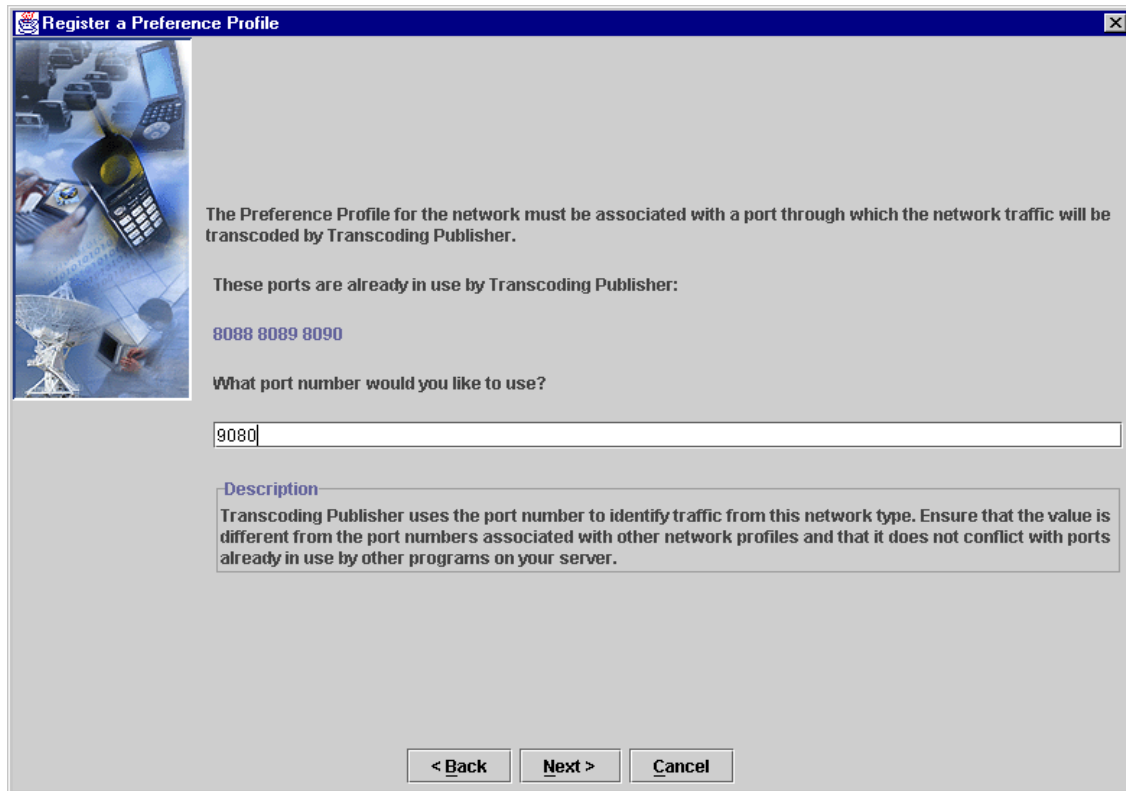


Figure 226. Assigning a port number with the network profile

We will see port numbers that IBM WebSphere Transcoding Publisher is already using. Enter the value for the port number you choose (9080 in our example). Be sure that the port you are choosing is available and then click **Next**.

Register a Preference Profile

Once the Preference Profile is registered, a name and description will make it easier to locate and identify in the Administration Console.

What would you like to name this preference profile?

New Network

How would you like to describe this Preference Profile?

This network profile is to show how to define a network profile

Description

This name will be used to refer to the Preference Profile when it is registered. You can use the name provided or you may change it.

This description can be used to further identify the Preference Profile. If a description has been provided, you can use it or you may change it.

< Back Next > Cancel

Figure 227. Naming the profile

We accept the values in the newnetwork.prop file. Notice that if you change the names here they will be changed in your file also. Click **Next**.

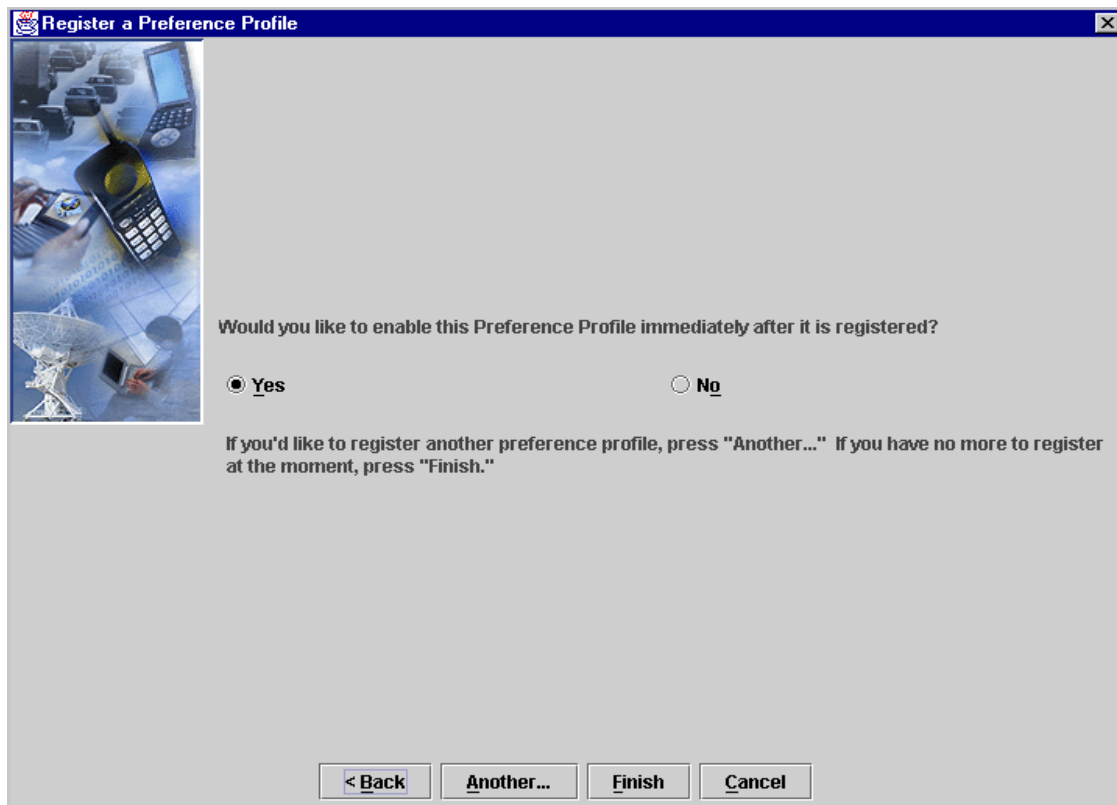


Figure 228. Accepting the registration

We can now either just register the profile or at the same time enable the profile. In our scenario, we enable the profile. Click **Finish**.

In the Administration Console we can see that our profile is registered.

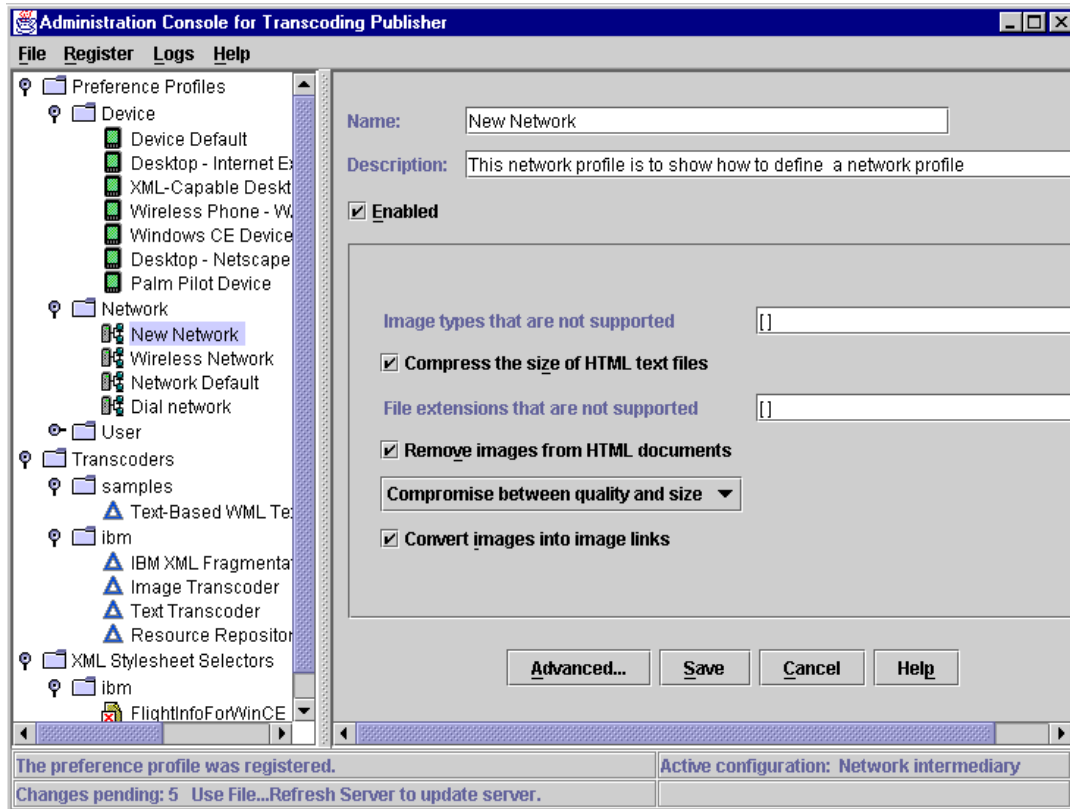


Figure 229. Administration Console showing New Network profile

Newnetwork.prop source:

```
#version = 1.0
#Wed Apr 26 17:28:36 EDT 2000
unsupportedImages=[ ]
compressSource=true
DescriptiveName=dadaadadadadada
unsupportedExtensions=[ ]
disposeImages=true
networkRule=9080
imageSizeQualityTradeoff=compromise
textLinksPreferredToImages=true
ConfigurableProperties=unsupportedImages{text} compressSource{bool}
unsupportedExtensions{text} disposeImages{bool}
imageSizeQualityTradeoff{comboBox(compromise,favorHighQuality,favorSmallSize)} textLinksPreferredToImages{bool}
Description=This network profile is to show how to define a network profile
```

12.4.2 Registering a device preference profile

From the Administration Console select **Register --> Preference Profile**.



Figure 230. Welcome window to register a profile

Click **Next**.



Figure 231. Browsing for the profile

Now we have to locate our new device profile. Click the **Browse** button.

We stored the profile in the *IBMTrans\etc\preferences\device* folder.

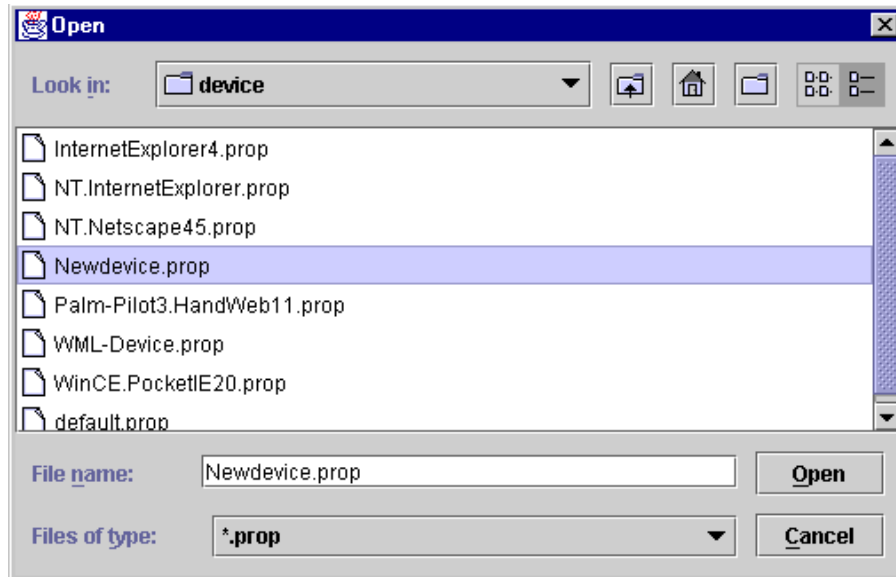


Figure 232. Selecting device profile

In our case, the profile is Newdevice.prop. Select the file and click **Open**.

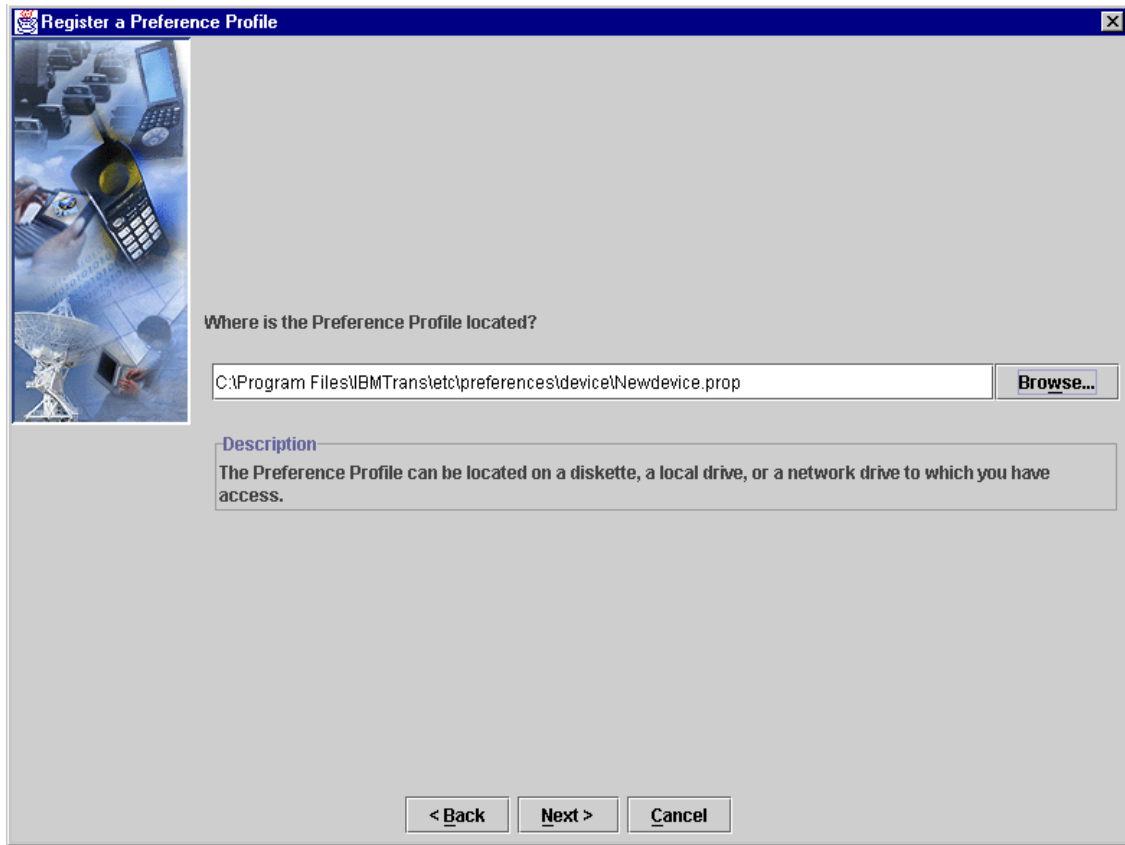


Figure 233. Registering a device preference profile

Another method is to type in the directory and the file name:

C:\Program Files\IBMTrans\etc\preferences\device\Newdevice.prop.

Click **Next**.

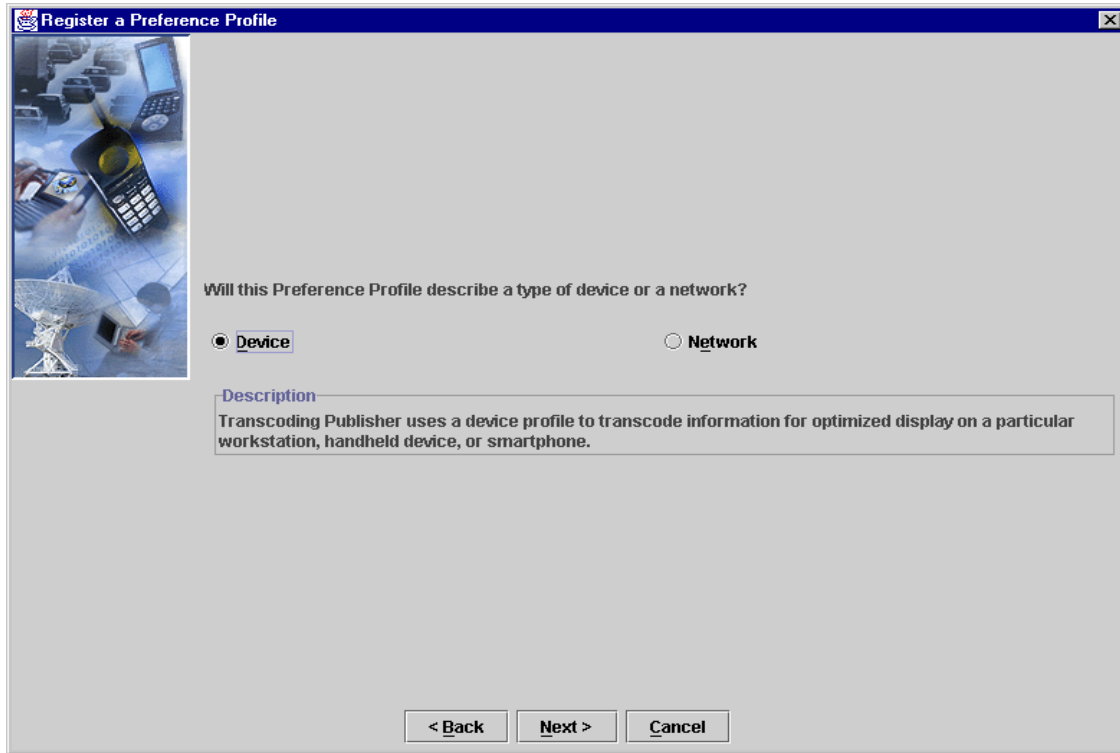


Figure 234. Selecting what profile to register

Select **Device** and click **Next**.

You will see the following window saying that the profile is already registered if you have stored the profile in the default network (see 12.4.1, “Registering network preference profile” on page 269). Just click **Yes**. If you select **No**, it will stop the registration process.



Figure 235. Duplicate profile confirmation

Register a Preference Profile

Once the Preference Profile is registered, a name and description will make it easier to locate and identify in the Administration Console.

What would you like to name this preference profile?

New Device

How would you like to describe this Preference Profile?

New Device profile to show to to create new device profile

Description

This name will be used to refer to the Preference Profile when it is registered. You can use the name provided or you may change it.

This description can be used to further identify the Preference Profile. If a description has been provided, you can use it or you may change it.

< Back Next > Cancel

Figure 236. Descriptive names for the profile

Next we have to register this device using a descriptive name. We recommend using the ones we defined when the profile was created. Notice that if you change these names here they will also be changed in the file.

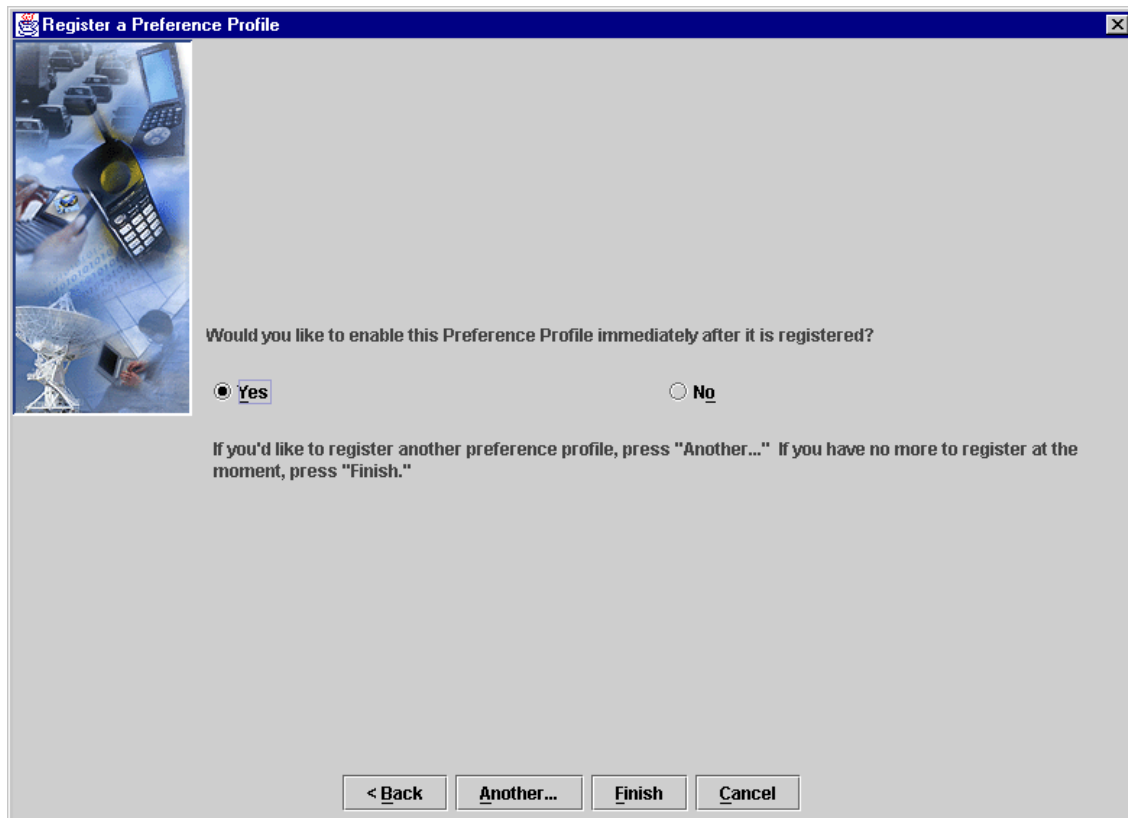


Figure 237. Finishing the registration

We want to enable the device immediately.

Click **Finish**.

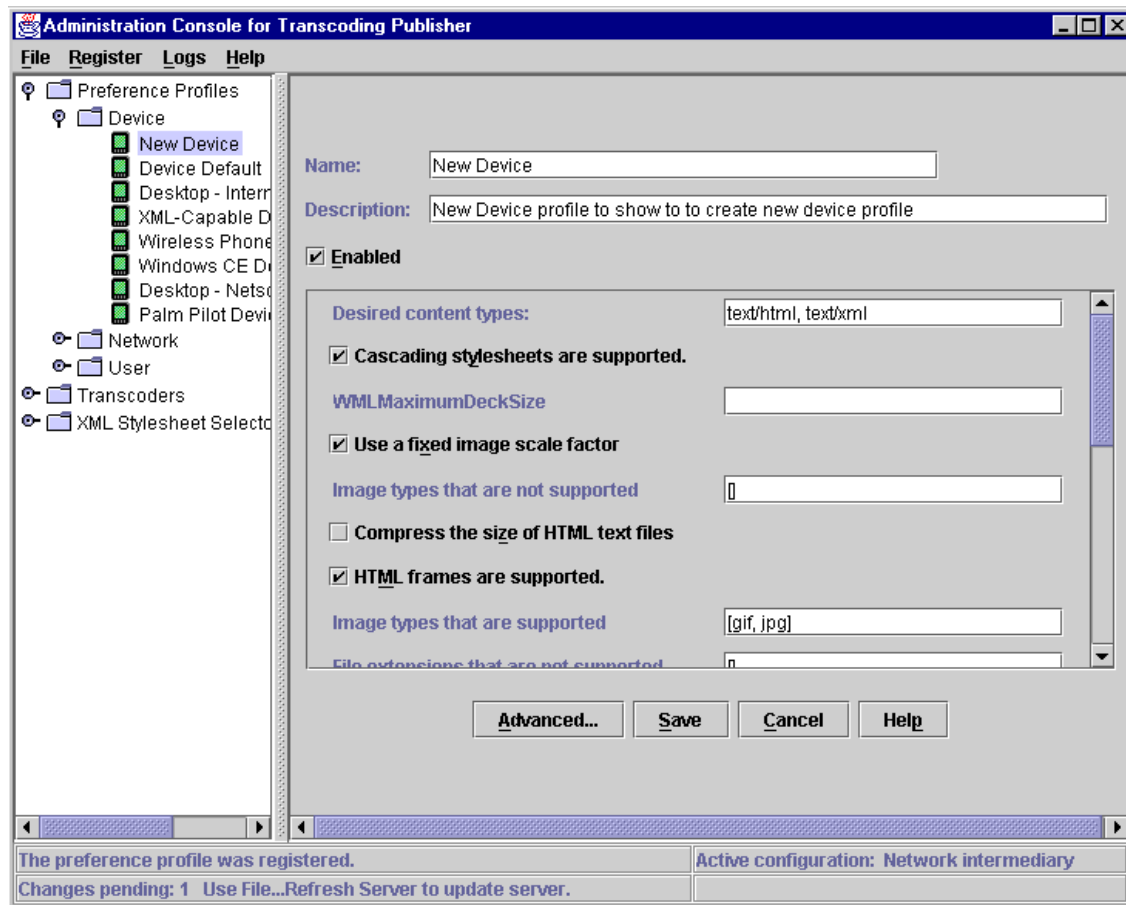


Figure 238. Administration Console showing our registered device profile

From the Administration Console we can check that our profile is registered and enabled.

Newdevice.prop source:

```
#version = 1.0
#Thu Apr 27 11:02:38 EDT 2000
desiredContentTypes=[text/html, text/xml]
cascadingStyleSheetsSupported=true
WMLMaximumDeckSize=
fixedImageScale=true
unsupportedImages=[]
compressSource=false
framesSupported=true
```

```

DescriptiveName=New Device
supportedImages=[gif, jpg]
unsupportedExtensions=[]
javaScriptSupported=true
screenCapability=high
clientSideImageMapsSupported=true
javaAppletsSupported=true
deviceRule=User_Agent%eWatch Browser V3.0
disposeImages=false
desiredDocumentTypeDefinitions=[]
ObjectsSupported=true
imageScaleFactor=0.8
convertTablesToUnorderedLists=false
imageSizeQualityTradeoff=compromise
textLinksPreferredToImages=false
colorSupported=true
ConfigurableProperties=desiredContentTypes{text}
cascadingStyleSheetsSupported{bool} WMLMaximumDeckSize{itext}
fixedImageScale{bool} unsupportedImages{text} compressSource{bool}
framesSupported{bool} supportedImages{text} unsupportedExtensions{text}
javaScriptSupported{bool} screenCapability{comboBox (high, medium, low)}
clientSideImageMapsSupported{bool} javaAppletsSupported{bool}
disposeImages{bool} desiredDocumentTypeDefinitions{text}
ObjectsSupported{bool} imageScaleFactor{itext}
convertTablesToUnorderedLists{bool}
imageSizeQualityTradeoff{comboBox (compromise, favorHighQuality, favorSmallSi
ze)} textLinksPreferredToImages{bool} colorSupported{bool}
Description=New Device profile to show how to create new device profile

```

12.4.3 List of preferences and their variables in profile files

In this release, the preference profile files can only be modified manually using any editor. So we will list here the variables and values for those files.

Table 5. List of preferences and associated variables

Preference	Variable
Desired content types	desiredContentTypes=[]
Cascading stylesheets are supported	cascadingStyleSheetsSupprted=true/false
WML maximum deck size	WMLMaximumDeckSize=
Use fixed image scale factor	fixedImageScale=true/false
Image types that are not supported	unsupportedImages= []

Preference	Variable
Compress the size of HTML text files	compressSource=true/false
HTML frames are supported	framesSupported=true/false
Image types that are supported	supportedImages= []
File extensions that are not supported	unsupportedExtensions= []
Java Script is supported	javascriptSupported=true/false
Device screen capability	screenCapability=high/medium/low
Image maps are supported	clientSideImageMapsSupported=true/false
Java applets are supported	javaAppletsSupported=true/false
Remove images from HTML files	disposeImages=true/false
Desired document type definitions	desiredDocumentTypeDefinitions=true/false
Objects supported	ObjectsSupported=true/false
Image scale	imageScaleFactor=
Convert tables to lists within lists	convertTablesToUnorderedLists=true/false
Image quality resolution	imageSizeQualityTradeoff=compromise/favorHighQuality/favorSmallSize
Convert images to image links	textLinksPreferredToImages=true/false
Color is supported	colorSupported=true/false
Detailed Description	Description=
Descriptive Name	DescriptiveName=

Chapter 13. Problem determination

In this chapter we provide a list of hints and tips for the most typical problems you will find when installing and implementing Transcoding Publisher in its different models (proxy, WAS filters and JavaBeans). We also include an overview of the log and trace facilities provided by WTP 1.1.

13.1 Typical problems

A list of common problems you may encounter while installing and setting up IBM WebSphere Transcoding Publisher follows:

1. Results received are not transcoded

- a. Cause 1: Client configurations:
 - Requests are not reaching WTP by passing the proxy. The results appear to not be transcoded because the request went directly to the Web server.
 - Check to ensure proxy settings have been specified and that the name or address is correct along with the appropriate port of WTP.
 - There is a setting on most browsers to bypass proxies for local addresses. If this option is enabled, requests for intranet pages will not go through the WTP proxy. To fix this, change the setting on the browser.
- b. Cause 2: Device profile does not exist or is not registered:
 - The device is not recognized or is not specified in WTP. In this case the default device profile is used. Verify that a device profile exists and is registered for the requesting device.

2. Request indicates that the Host is not found

- a. Cause: WTP cannot reach the host in the request.
 - Check WTP server settings to ensure that a valid socks or proxy entry exists to allow WTP to reach destinations outside the firewall.

3. Results received are XML and have not been transcoded

- a. Cause: stylesheet definitions:
 - 1. Cause 1: The XML document that was received did not have a registered stylesheet.
 - 2. Cause 2: Registered stylesheets do not exist that will generate the *Content Type* required by the requesting device.

3. Cause 3: The document name is incorrect in the stylesheet registration.
 - Resolution: Review stylesheets and correct configuration items or create and register a new stylesheet. Note that in instances where a stylesheet match is not found the unmodified XML document will be returned to the client.

4. XML received and not transcoded, Servlet Model

- a. Cause 1: *TranscodingFilter* servlet is not registered in WAS.
- b. Cause 2: *TranscodingFilter* servlet is not registered as a MIME filter for “text/xml” in WAS.
- c. Cause 3: *TranscodingPreamble* servlet was not specified in the “alias” specifications for the data generation servlet.
 - Resolution: Review your servlet configuration data.

Note: Any of the points listed in item 3 can also cause this problem.

5. WML Errors

- The majority of these errors will be found in the HTML to WML conversions. The errors generally come from malformed HTML code which cannot be properly parsed by Transcoder Publisher.
- The first course of action for these types of problems is to attempt to view the WML in a system where the error in the WML deck can be found, for example, the Nokia phone simulator toolkit. Corrective action may include correcting the HTML source if possible. If correcting the source is not possible then (1) consider a “clipper” to fix the problem and (2) collect the pertinent information and report the problem.
- If a document is not being transcoded to WML properly, one of the first things to verify is which preference profiles were used to select transcoders. When a document is processed for WML, a directory is created in etc/resource repository/, with the name of the device profile selected to process this document. Temporary files are cached within this directory. One way to determine which device profile was in use is to look for a directory in etc/resource repository that has the name of a device profile.

6. Transcoded result is not tailored to the Palm browser or WML browser

- Cause: The generic transcoding services for HandWeb in the Palm Pilot and HTML-to-WML conversion for WAP phones may not create the most usable view on the receiving device. This is due to the constraints of the smaller screens, richness of content in the HTML

source, and potentially the removal of JavaScript operations from the HTML source.

- Resolution: review transcoding parameters for the Palm device to see if modification of the settings improves the appearance. For both the Palm and WML devices consider the creation of site-specific “clippers” to tailor the responses for the specific device environment.

7. Refresh server function stops the system

- Cause: On some Windows NT and Windows 2000 systems, the Refresh server function has been found to cause the CPU to run at 100%, effectively stopping the system. The Refresh Server function is also incompatible with some Virtual Private Networks (VPNs).
- Resolution: If you experience this CPU problem, or if you use a VPN, you can disable the Refresh server function to eliminate the problem. The enablement of server refreshing is specified in the Transcoding Publisher ENVIRONMENT.prop file. The file governing your Transcoding Publisher environment depends on your configuration:

- Simple proxy: etc/config/proxy/ENVIRONMENT.prop
- Caching proxy: etc/config/enterprise/ENVIRONMENT.prop
- Servlet: etc/config/servlet/ENVIRONMENT.prop

The ENVIRONMENT.prop file defines a variable called EnableRMI, which governs the enablement of the Refresh Server function. To disable server refresh, set this to:

EnableRMI = false

If you make configuration changes after disabling the Refresh Server function, you must restart the Transcoding Publisher server to implement the configuration changes. To restart the server on Windows NT or Windows 2000, use the Services applet on the Control Panel.

13.2 Tools

The following is a list of tools provided by IBM WebSphere Transcoding Publisher to help you in problem determination:

1. Request Viewer

For the proxy model use the Request Viewer in the Transcoding Publisher toolkit to view the connections and for debugging.

2. Transcoding Publisher Trace

The Transcoding Publisher Trace provides a full detailing of the operations of WTP. Reviewing the trace can in many instances help identify the source of a specific problem.

3. WebSphere Application Server Console

- Using the WAS console with tracing enabled will allow viewing of the processing of the TranscodingPreamble and TranscodingFilter servlets to ensure they are being invoked properly.
- The WAS debug console can also be used for viewing which servlets are active in the system. Therefore, it can be used to verify that WTP servlets TranscodingPreamble and TranscodingFilter are running when the content generation servlet runs. In addition, some early errors that cause the filter not to work can be easily viewed with this tool, and so it can be helpful in diagnosing startup problems.

4. XML4J and LotusXSL Engines

- Stylesheet content validation can be accomplished using these tools from the IBM XML services. For more information, see

<http://www.ibm.com/xml>

13.3 Logging and tracing

This section shows how to enable logging and tracing, change their properties, and view trace and message logs. Both facilities allow you to determine the proper operation of WTP. They also help you in problem determination to troubleshoot and isolate problems related to WTP transcoding.

The Logs menu entry in the Administration Console is used to deal with message and trace logging. Both can be enabled and disabled from this menu, their properties set and their viewers started:

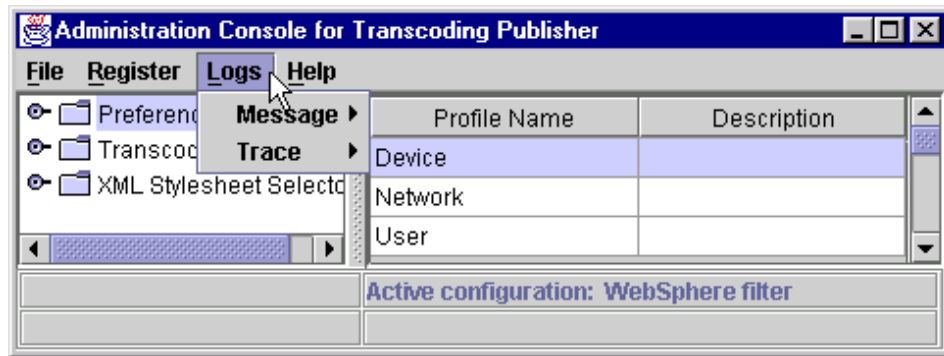


Figure 239. The Administration Console Logs menu

13.3.1 Logging

A logging facility is provided with IBM WebSphere Transcoding Publisher. Use the Logs menu in the Administration Console to:

- Enable or disable message logging.
- Modify properties of message logging.
- View the current message log file.

13.3.1.1 Enable message logging

As shown in Figure 240, check the Enabled box to enable message logging.

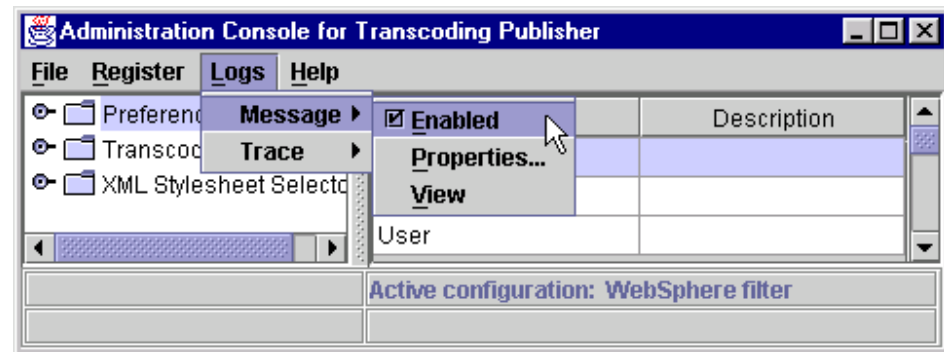


Figure 240. Administration Console - enable message logging option

13.3.1.2 Message Properties panel

Use the Message Properties Panel to select the type of messages to be logged. Any combination of information, warning and error messages can be selected.



Figure 241. Message log properties - categories

When message logging is enabled without any message categories selected it is equivalent to having message logging disabled, except that performance might possibly be worse.

The following categories can be selected:

- **Information:** records normal events. Use these messages to determine whether expected operations occur, for example, to verify that a configuration change takes place as expected.
- **Warning:** indicates a possible problem. Request may still be handled successfully.
- **Error:** indicates a definite problem requiring administrator attention. Requests will probably not be handled successfully.

By default, all categories of messages are logged. The message log file resides in the /log directory and is named **TranscoderMessages.log**. The file is not automatically deleted or restarted. You can view the message file by selecting **View** under **Message** on the **Logs** menu.

13.3.1.3 The Message Viewer

This tool is a very simple viewer without any sophisticated features. To do any real work with the message files you will probably have to use a full function editor.

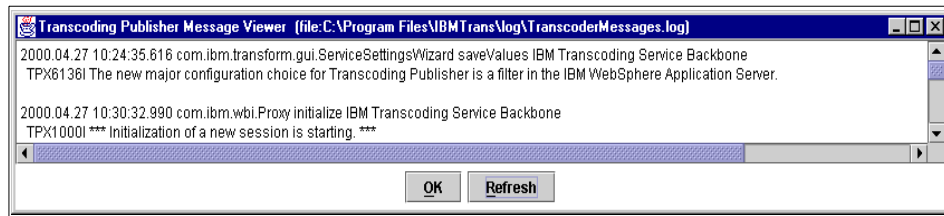


Figure 242. WTP Message Viewer

In addition, the Message Viewer may not always work correctly with double-byte character set languages. The messages may look garbled in some cases. However, this is only a display problem as the actual messages are properly stored in the message log file. The message log file can be viewed using an editor supporting the specific text in that language.

For example, Figure 243 shows the use of an editor to browse the TranscoderMessage.log file.

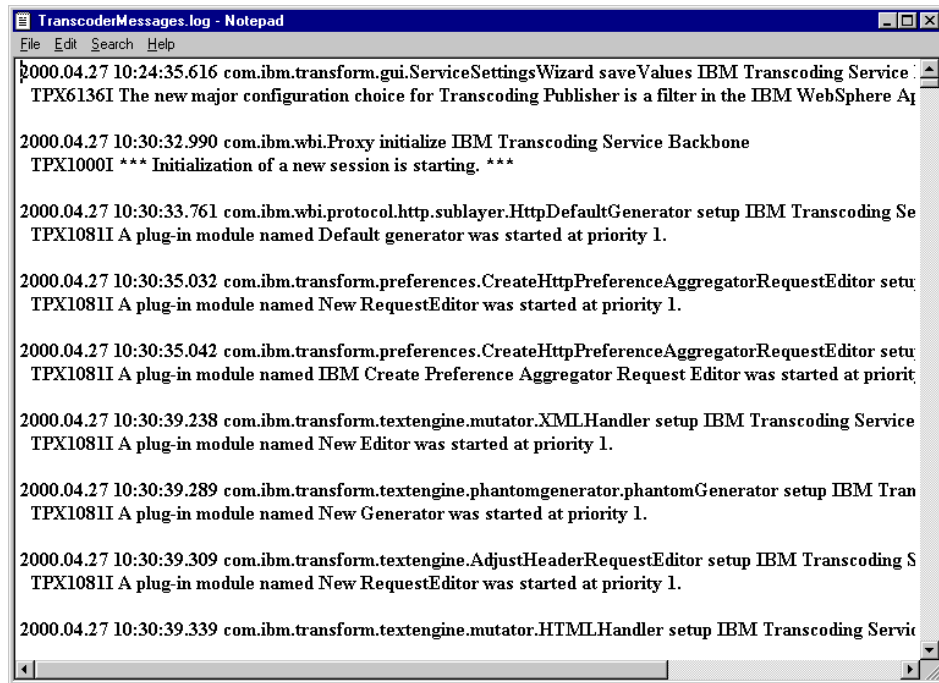


Figure 243. Using an editor to browse the message log (TranscoderMessage.log)

The Message Viewer also shows informational messages indicating that the configuration choice for Transcoding Publisher is, for example, a filter, that the proxy has been initialized successfully, and other related information as illustrated in Figure 244.

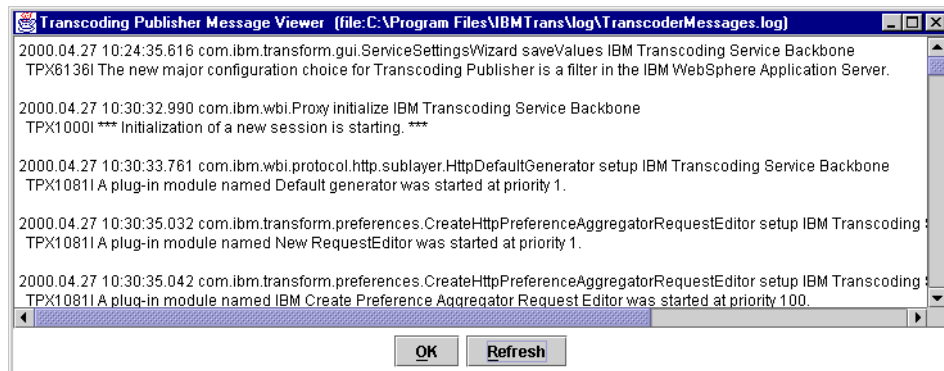


Figure 244. Message Viewer - sample informational messages

13.3.2 Tracing

The trace facility in IBM WebSphere Transcoding Publisher provides you with the required information to analyze potential problems. However, there is a performance cost when setting any level of tracing. Moreover, in cases where performance is being measured or compared, you should turn off all tracing options to avoid delays and performance overhead.

Note

When you install Transcoding Publisher, the trace function is turned on at the lowest setting. This will enable you to gather information on the installation and configuration processes in case you have any problems. When you have completed the configuration process, turn tracing off. This will improve the performance of Transcoding Publisher. To turn tracing off, select **Logs->Trace** and deselect the **Enabled** checkbox.

However, during problem determination, tracing is very helpful and it should be turned on when the product is being installed and configured, or when the configuration is being altered. Any exceptions caused by configuration problems will be captured in the trace log. Without this information, it will probably be very difficult to diagnose problems.

From the Logs menu, you can:

- Enable or disable the trace.
- Modify properties of the trace.
- View the current trace file.

13.3.2.1 Enable traces

As shown in Figure 245, check the Enabled box to enable trace logging.

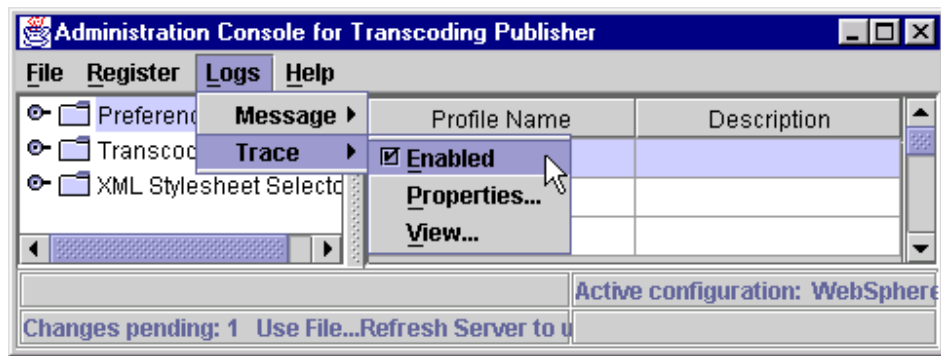


Figure 245. Administration Console - enable trace option

13.3.2.2 Trace Properties panel

Use the Trace Properties panel to specify the categories of trace messages that will be logged and the number and size of trace files that will be used. Tracing affects the performance of Transcoding Publisher and is usually performed only at the request of a service representative.

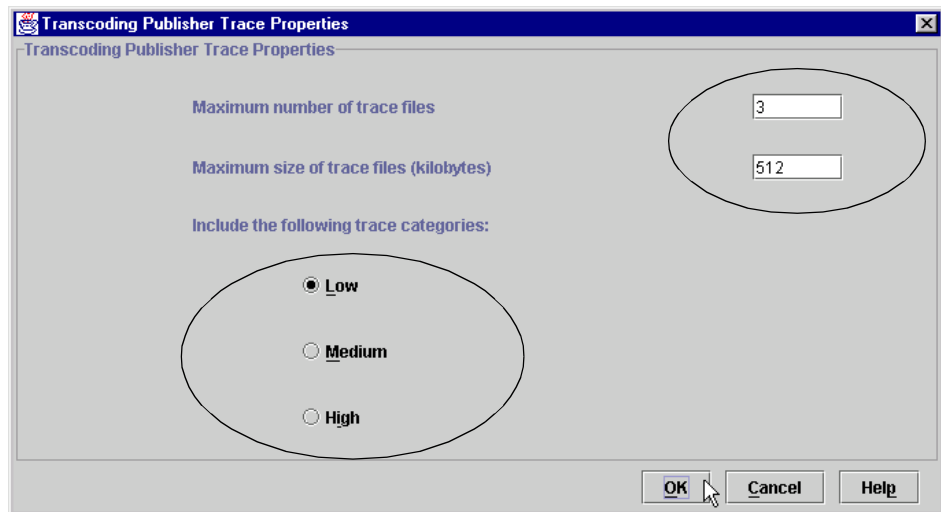


Figure 246. Administration Console - setting trace properties

As shown in Figure 246, the following options can be configured in the trace properties panel:

- **Trace files:** You can specify how many trace files can be created and the maximum size (per file, not total). Files are created in the /log directory.

The first file created is named TranscoderTrace1.log. When the maximum size is reached, this file will be renamed TranscoderTrace2.log and a new TranscoderTrace1.log file created for new messages.

TranscoderTrace1.log is always the newest file.

When the maximum number of files have been filled, the oldest file will be deleted, the suffix number of each remaining file will be increased by one, and a new TranscoderTrace1.log will be created for new messages. A new file is begun when Transcoding Publisher is restarted.

- **Trace levels:**

- **Low.** It includes error messages and external API calls and it may cause some performance degradation.
- **Medium.** It includes low messages plus all trace records, but in less detail than the high option. It may cause moderate performance degradation.
- **High.** It includes low and medium messages plus extensive intermediate results of operations. It may cause serious performance degradation. In general, it is typically used for one or two requests at the most.

Note: When WTP is installed, traces are set to “low”. It is recommended that you run this level of traces during development work. However, traces should be disabled during production or during performance profiling.

13.3.2.3 Trace Viewer

Like the message viewer, in IBM WebSphere Transcoding Publisher Version 1.1, the trace viewer (see Figure 247) is a simple tool without any sophisticated features.

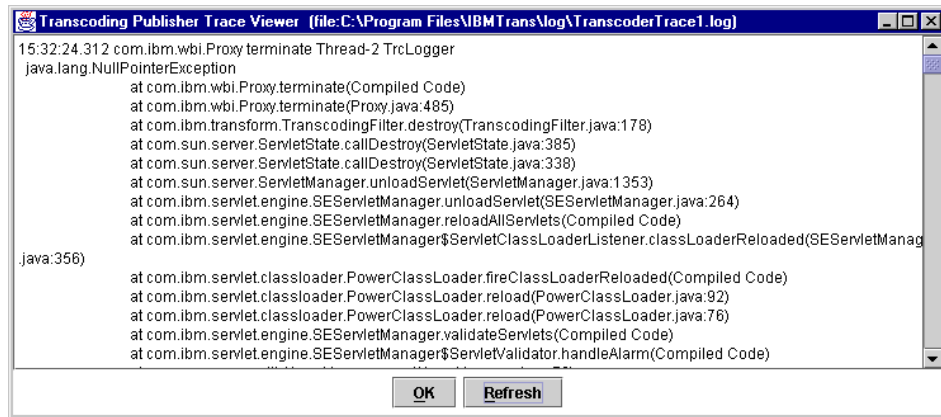


Figure 247. Administration Console - Trace Viewer

In some cases, if you want to do any advanced functions (such as browse and find), with the message trace files, you should consider using a full function editor as illustrated in Figure 248.

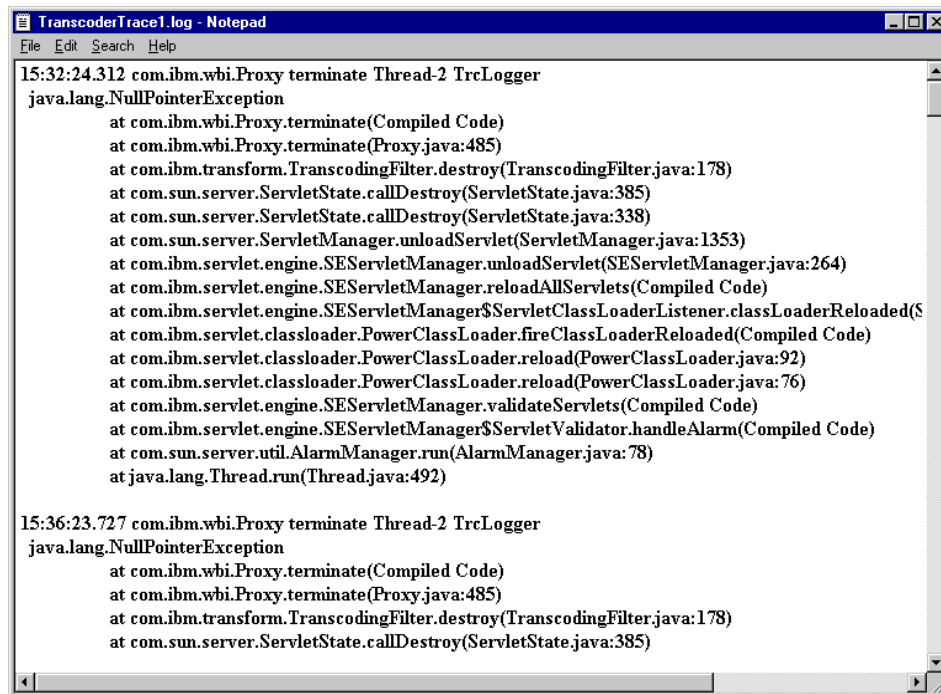


Figure 248. Using an editor to browse the trace log (TranscoderTrace1.log)

The double-byte problem does not occur with the trace viewer because all the trace records are in English.

Among other information, the trace file shows the complete Java stack trace for an exception condition. This can be very helpful to development when debugging problems. In addition, WAS trace and log files may also be useful in diagnosing problems, especially if they appear at WTP startup. Since WTP information cannot be sent to the trace files until the service is running, some early diagnostics go to the WAS files.

13.4 Reporting problems

The following process is used by customers to report problems to the IBM Support Center for the IBM WebSphere Transcoding Publisher product:

- If you live in the United States or Canada, you may call: 1-800-237-5511.
- Via the Internet on the Transcode support page:
<http://www.ibm.com/software/webservers/transcoding/support>.

Appendix A. TranscodingSamples sample program

TranscodingSamples is a sample program provided by IBM WebSphere Transcoding Publisher Version 1.1 in the toolkit subdirectory. The program produces text content (for example, XML) based on the file extension passed as a parameter. TranscodingSamples can be run either as a servlet in a WAS environment or as a MEGLet in the WTP proxy model. It provides the same service in both cases.

It is used in this redbook (Chapter 7, "Transcoding with WAS filters" on page 93) to demonstrate WTP transcoding running as WAS filters.

```
/*
*****
/*****
* Licensed Materials - Property of IBM
* IBM Transcoding Publisher Toolkit
* (c) Copyright IBM Corp. 2000 All rights reserved.
*
* US Government Users Restricted Rights - Use, duplication or
* disclosure restricted by GSA ADP Schedule Contract with IBM
* Corp.
*
* DISCLAIMER OF WARRANTIES. The following [enclosed] code is
* sample code created by IBM Corporation. This sample code is
* not part of any standard or IBM product and is provided to you
* solely for the purpose of assisting you in the development of
* your applications. The code is provided "AS IS", without
* warranty of any kind. IBM shall not be liable for any damages
* arising out of your use of the sample code, even if they have
* been advised of the possibility of such damages.
*
* TranscodingSamples.java
*
*****
*/

/**
 * TranscodingSamples serves up sample files provided by the Content Magic Toolkit.
 * For security reasons, only files listed in toolkit/RequestableSamples.prop
 * will be delivered by this servlet.
 *
 * Usage:
 *      http://host/servlet/TranscodingSamples?name=key
 *
 * Where key is listed in toolkit/RequestableSamples.prop
 *
 * InitParms:
 *
 *      InstallPath = The path to the root of the CM install tree (e.g., /opt/IBMagic)
 *                      Defaults to ".".
 *      LogRequests = "true" to generate a log of requested files. Default is no log.
 */

import java.io.*;
import java.util.Properties;
import javax.servlet.*;
import javax.servlet.http.*;
```

```

public class TranscodingSamples extends HttpServlet
{
    private final static String TOOLKIT_HOME = "toolkit";
    private final static String REQUESTABLE_SAMPLES = TOOLKIT_HOME + File.separator +
"RequestableSamples.prop";
    private String installPath = ".";
    private boolean logRequests = false;
    private Properties samples = null;

    // ----- init -----

    public void init(ServletConfig config) throws ServletException
    {
        super.init(config);

        // Work through the init parameters.

        String temp = getInitParameter("InstallPath");
        if (temp != null)
        {
            installPath = temp;
        }

        temp = getInitParameter("LogRequests");
        if (temp != null && temp.equalsIgnoreCase("true"))
        {
            logRequests = true;
        }

        // And then get the list of samples we'll accept.

        String tableName = installPath + File.separator + REQUESTABLE_SAMPLES;
        samples = new Properties();
        try
        {
            samples.load(new FileInputStream(new File(tableName)));
        }
        catch (Exception e)
        {
            getServletContext().log(e, "TranscodingSamples: Unable to load " + tableName); //
JSDK 2.0 backward compat
        }
    }

    // ----- doGet -----

    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        // Get the name of the file (query parameter "name")

        String name = request.getParameter("name");
        if (name == null)
        {
            response.sendError(HttpServletResponse.SC_BAD_REQUEST);
            return;
        }

        // Map it to the real file name & get the content type

        String file = samples.getProperty(name);
        if (file == null)

```

```

{
    response.sendError(HttpServletResponse.SC_FORBIDDEN);
    return;
}
String document = installPath + File.separator + TOOLKIT_HOME + File.separator +
    file.replace('/', File.separatorChar);
String contentType = getServletContext().getMimeType(document);
if (contentType == null)
    contentType = "text/plain";
contentType = contentType.trim();          // Required on some versions of WAS
if (logRequests)
    log("TranscodingSamples: Serving file " + document + ", type = '"
        + contentType + "'");

// Attempt to return the file.

FileInputStream in = null;
try
{
    response.setContentType(contentType);
    byte buffer [] = new byte [4096];
    int bytesRead;
    in = new FileInputStream(document);
    OutputStream out = response.getOutputStream();

    while ((bytesRead = in.read(buffer)) >= 0)
        out.write(buffer, 0, bytesRead);
}
catch (SecurityException e)
{
    response.sendError(HttpServletResponse.SC_FORBIDDEN);
}
catch (Exception e)
{
    getServletContext().log(e, "TranscodingSamples: error returning " + document);
    response.sendError(HttpServletResponse.SC_NOT_FOUND); // maybe...
}
finally
{
    if (in != null)
        in.close();
}
}
}

```

Appendix B. XML stylesheet (FlightInfoForNet_IE.xsl)

In this appendix we list the XML stylesheet used in 7.5, “Sample Scenario: transcoding XML content” on page 113.

```
<?xml version="1.0"?>

<!-- Change Flight Information from Flights DTD to HTML for Netscape/IE -->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/">
    <html>
      <title> Flight Info For HTML </title>
      <body>
        <xsl:text>
</xsl:text>

          <p>
<xsl:text>
</xsl:text>

          <xsl:text>
</xsl:text>

          <h1>
            <xsl:text> Flight Info</xsl:text>
          </h1>
          </p>
          <p>
            <table border="2">
              <xsl:apply-templates/>
              <xsl:text>
</xsl:text>

            <xsl:text>
</xsl:text>

            </table>
          </p>

          <xsl:text>
</xsl:text>

          </body>
        </html>
      </xsl:template>

      <xsl:template match="Flights">
        <xsl:if test="AirCarrier">

          <tr>
            <td>
              <b><xsl:text>Airline: </xsl:text></b>
            </td>
            <td>
              <xsl:value-of select="AirCarrier"/>
            </td>
          </tr>

        </xsl:if>
        <xsl:if test="FlightNumber">

          <tr>
            <td>
              <b><xsl:text>Flight: </xsl:text></b>

```

```

        </td>
        <td>
        <xsl:value-of select="FlightNumber"/>
        </td>
        </tr>

</xsl:if>

<xsl:text> </xsl:text>

<xsl:text> </xsl:text>
<xsl:for-each select="FlightSegment">

    <xsl:value-of select="DepartingInfo/City"/>
    <xsl:value-of select="ArrivingInfo/City"/>
    <xsl:text>
</xsl:text>

        <tr>
        <td>
        <b><xsl:text>Departure: </xsl:text></b>
        </td>
        <td>
        <xsl:value-of select="DepartingInfo/Scheduled"/>
        </td>
        <td>
        <xsl:value-of select="DepartingInfo/Status"/>
        </td>
        <td>
        <xsl:value-of select="DepartingInfo/Time"/>
        </td>
        </tr>

        <xsl:if test="DepartingInfo/Terminal">

            <tr>
            <td>
            <b><xsl:text>Terminal: </xsl:text></b>
            </td>
            <td>
            <xsl:value-of select="DepartingInfo/Terminal"/>
            </td>
            </tr>

        </xsl:if>

        <tr>
        <td>
        <b><xsl:text>Gate </xsl:text></b>
        </td>
        <td>
        <xsl:value-of select="DepartingInfo/Gate"/>
        </td>
        </tr>

        <tr>

```



```

        <td>
        <b><xsl:text>Arrival: </xsl:text></b>
        </td>
        <td>
        <xsl:value-of select="ArrivingInfo/Scheduled"/>
        </td>
        <td>
        <xsl:value-of select="ArrivingInfo/Status"/>
        </td>
        <td>
        <xsl:value-of select="ArrivingInfo/Time"/>
        </td>
    </tr>

    <tr>
    <td>
    <b><xsl:text>Terminal: </xsl:text></b>
    </td>
    <td>
    <xsl:value-of select="ArrivingInfo/Terminal"/>
    </td>
    </tr>

    <tr>
    <td>
    <b><xsl:text>Gate: </xsl:text></b>
    </td>
    <td>
    <xsl:value-of select="ArrivingInfo/Gate"/>
    </td>
    </tr>
</xsl:for-each>
</xsl:template>
</xsl:stylesheet>

```

Appendix C. CSD Service Fix Pack - APAR IC26798

The sample scenarios described in this redbook have been implemented using this service fix pack. The following problems have been fixed in this CSD via APAR IC26798:

- Fixes to server notification of configuration changes
- Meglets do not run when running as an NT service
- ImageTranscoder.jsp and ImageTranscoder10.jsp need query string
- TranscodingSamples running as editor, not generator
- Performance improvements
- HTML->WML transcoding fixes
- Cache support fixes
- WTP hangs on heavy WAP traffic
- Device rules creation does not handle blank space in name
- Message viewer DBCS support
- Double-byte problem for link/image translation
- BIDI support
- AIX install: do not add RunTranscoding to inittab more than once
- Mark acread executable for AIX and SUN on CD
- Tracing improvements
- Administration console usability improvements and fixes
- Migrate XSL to Xalan 1.0
- WML fragmentation fixes (navigation, reliability)
- Bad image link created when going through WebSphere

Appendix D. Special notices

This publication is intended to help network specialists and application developers to install, configure and enable transcoding in Web applications using the IBM WebSphere Transcoding Publisher Version 1.1 product. The information in this publication is not intended as the specification of any programming interfaces that are provided by the IBM WebSphere Transcoding Publisher Version 1.1. See the PUBLICATIONS section of the IBM Programming Announcement for IBM WebSphere Transcoding Publisher Version 1.1 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been

reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AFP	AIX	AS/400
CICS	DB2	IBM
Lotus	MQSeries	Notes
Netfinity	OS/390	RS/6000
S/390	SecureWay	SP
System/390	VisualAge	WebSphere
Wizard	WorkPad	

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Appendix E. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

E.1 IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 315.

- *IBM WebSphere Performance Pack Usage and Administration*, SG24-5233
- *IBM WebSphere and VisualAge for Java Database Integration with DB2, Oracle, and SQL Server*, SG24-5471
- *WebSphere Application Servers: Standard and Advanced Editions*, SG24-5460
- *Building Integration Objects with IBM SecureWay Host Publisher Version 2.1*, SG24-5385
- *IBM WebSphere Performance Pack: Caching and Filtering with IBM Web Traffic Express*, SG24-5859

E.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at ibm.com/redbooks for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
IBM System/390 Redbooks Collection	SK2T-2177
IBM Networking Redbooks Collection	SK2T-6022
IBM Transaction Processing and Data Management Redbooks Collection	SK2T-8038
IBM Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
IBM AS/400 Redbooks Collection	SK2T-2849
IBM Netfinity Hardware and Software Redbooks Collection	SK2T-8046
IBM RS/6000 Redbooks Collection	SK2T-8043
IBM Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

E.3 Other resources

These online publications, shipped with the product, are also relevant as further information sources:

- *Administrator's Guide*
- *Readme file*
- *JDK 1.1.8 Toolkit Readme file*
- *Host Publisher User's Guide*

E.4 Referenced Web sites

Find frequently asked questions (FAQs), white papers, and additional information at the product Web site:

- <http://www.ibm.com/software/webservers/transcoding>

These Web sites are also relevant as further information sources:

- IBM Host Publisher on the Internet:
<http://www.ibm.com/software/network/hostpublisher>
- WapForum on the Internet:
<http://www.wapforum.org>
- Nokia WAP Phone Emulator:
<http://www.nokia.com/wap/index.html>
- WinGate on the Internet:
<http://www.wingate.com>

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** <http://www.redbooks.ibm.com/>

Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States or Canada	pubscan@us.ibm.com
Outside North America	Contact information is in the "How to Order" section at this site: http://www.elink.ibm.link.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the "How to Order" section at this site: http://www.elink.ibm.link.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the "How to Order" section at this site: http://www.elink.ibm.link.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at <http://w3.itso.ibm.com/> and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at <http://w3.ibm.com/> for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name	Last name
------------	-----------

Company

Address

City	Postal code	Country
------	-------------	---------

Telephone number	Telefax number	VAT number
------------------	----------------	------------

<input type="checkbox"/> Invoice to customer number	
---	--

<input type="checkbox"/> Credit card number	
---	--

Credit card expiration date	Card issued to	Signature
-----------------------------	----------------	-----------

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

Administration Console. The Administration Console (graphical interface) is a Java application that is used to perform configuration, administration and tracing tasks, as well as integrating and configuring new profiles and transcoders.

Applet. A Java applet is a small application program that is downloaded to and executed on a Web browser or network computer. A Java applet typically performs the type of operations that client code would perform in a client/server architecture. It edits input, controls the screen, and communicates transactions to a server, which in turn performs the data or database operations.

API. Application Program Interface.

Bean Managed Persistence. Bean Managed Persistence (BMP) is a term used to describe a type of entity EJB where the bean developer specifies how the bean is to be persisted to a database by writing Java code in the appropriate methods to perform the tasks required.

Cache Server. Some networks use a cache server to store Web pages and other data, so that if the same pages are requested frequently, they can be served from the cache rather than repeatedly retrieved from external Web servers. The external cache is an HTTP proxy such as IBM Web Traffic Express. IBM WebSphere Transcoding Publisher can use it to store and retrieve transcoded Web pages and intermediate results to avoid repeating the transcoding of frequently accessed pages and giving thus better performance.

CICS. Customer Information Control System is an IBM application enabler system very popular in mainframes.

Component Broker. Component Broker is an IBM CORBA management server product that provides an object request broker (ORB) to facilitate the deployment of CORBA objects. The EJB deployment engine in WebSphere is based

largely on similar services in Component Broker. See <http://www.software.ibm.com/ad/cb> for more information.

Connectors. The term connectors, or e-business connectors, is used to describe gateway products from IBM that allow access to enterprise data on back-end systems over the Internet. They include direct browser access to back-end systems such as DB2 through Net.data and also Java access through products such as the CICS gateway for Java. See <http://www.software.ibm.com/ebusiness/connectors.html> for more information.

Container Managed Persistence. Container Managed Persistence (CMP) is a term used to describe a type of entity EJB where the code to persist the bean to a database is generated at deployment time by the EJB container.

Clustering. Clustering is a technique used to provide scalability through the use of multiple copies of an application on the same or separate machines. Careful management of the different applications is necessary to ensure that they work together effectively. WebSphere has limited clustering support in Version 2.x and more support in Version 3.0.

CORBA. Common Object Request Broker Architecture (CORBA) is a cross-platform, industry-standard distributed object protocol. CORBA is used to locate and use objects on a variety of platforms, written in a variety of languages across a network. See <http://www.omg.org> for more information on CORBA.

e-business. e-business is a term used by IBM to describe the use of Internet technologies to transform business processes. What this means in practice is using Internet clients such as Web browsers as front ends for applications that access back-end legacy systems to allow greater access. See <http://www.software.ibm.com/ebusiness> for more information.

Enterprise Java Beans. Despite the name, Enterprise Java Beans or EJBs are not Java Beans. Enterprise Java Beans are server-side Java components that are designed for distributed environments. They do not exist in isolation but are rather deployed in containers that provide services such as security, naming and directory services and persistent storage. WebSphere Application Server is just such a container. See

<http://java.sun.com/products/ejb/> for more information.

IBM WebSphere Transcoding Publisher. IBM WebSphere Transcoding Publisher is a network software that modifies content presented to users based on the information associated with the request, such as device constraints, network constraints, user preferences, and organizational policies. Transforming content can reduce or eliminate the need to maintain multiple versions of data or applications for different device types and network service levels.

Image Transcoder. Image Transcoder is the transcoder that can scale, modify quality, and modify color levels in JPEG and GIF images. Additionally, the Image Transcoder can convert JPEGs to GIFs for devices that do not render JPEGs.

Intermediary. In a typical Web environment, information is simply sent from a server to a browser for display and interaction. However, there are many ways that adding an intermediary between the browser and the server can improve the system. For example, an intermediary can keep track of the information the user has viewed in order to make it easier to find information again. Or, an intermediary may enhance the information that the user sees by adding annotations and personalization beyond what the server was designed to do. Intermediaries turn the network into a “smart pipe” with applications that can enhance the information on the Web.

IIOP. Internet Inter ORB Protocol (IIOP) is an internet protocol used for CORBA object communication. For more information see <http://www.what-is.com/iiop.htm>.

Java Application. A Java application is a program written in Java that executes locally on a computer. It allows programming operations in addition to those used in applets which can make the code platform dependent. It can access local files, create and accept general network connections, and call native C or C++ functions in machine-specific libraries.

JavaBeans. JavaBeans are Java components designed to be used on client systems. They are Java classes that conform to certain coding standards. They can be described in terms of their properties, methods and events. JavaBeans may be packaged with a special descriptor class called a BeanInfo class and special property editor classes in a JAR file. Java Beans may or may not be visual components. See <http://www.javasoft.com/beans/docs> for more information.

JavaServer Pages. JavaServer Pages are HTML source files that include Java extensions to provide dynamic content and increased functionality. JavaServer Pages are compiled into Servlets before deployment. See <http://www.software.ibm.com/ebusiness/pm.html#JavaServer> Pages.

JDBC. JDBC is a Java API that allows Java programs to communicate with different database management systems in a platform-independent manner. Database vendors provide JDBC drivers for their platforms that implement the API for their database, allowing the Java developer to write applications to a consistent API no matter which database is used.

JNDI. Java Naming and Directory Interface (JNDI) is an API that allows Java programs to interface and query naming and directory services in order to find information about network resources. JNDI is used in WebSphere to provide a directory of Enterprise Java Beans. See <http://java.sun.com/products/jndi/index.html> for more information.

JSP. JavaServer Pages.

MEG. A plug-in encapsulates a correlated set of transcoding components (MEGs). A MEG is one of Monitor, Editor (Request or Document) or Generator. More information on MEGs and WBI (Web Intermediaries) can be found from <http://www.almaden.ibm.com/cs/wbi>.

PDA. Personal Digital Assistant.

Persistence. Persistence is a term used to describe the storage of objects in a database to allow them to persist over time rather than being destroyed when the application containing them terminates. Enterprise Java Bean containers such as WebSphere provide persistence services for EJBs deployed within them.

Preference Aggregator. Preference Aggregator is the process where the client's device, network type, and any other information that might cause the transcoders to produce a different variant (or permutation) of the resource, are determined.

Preference Aggregation. Preference aggregation is the process where the client's device, network type, and any other information that might cause the transcoders to produce a different variant (or permutation) of the resource, are determined.

Preference profiles. IBM WebSphere Transcoding Publisher uses preference profiles to represent the characteristics of devices and networks, and a default user profile to represent organizational policies. Each profile defines IBM WebSphere Transcoding Publisher how to treat documents that will be delivered to that device or over that network.

A preference profile can represent a particular type of device, such as a WorkPad, or a particular network type, such as a wireless network.

Proxy. Transcoding Publisher connects through a proxy server that is configured with a firewall to manage network traffic and to protect your network from outside intrusion.

RMI. Remote Method Invocation (RMI) is a lightweight distributed object protocol that allows Java objects to call each other across a network. RMI is part of the core Java specification. See

<http://java.sun.com/products/jdk/rmi/index.html> for more information.

Scalability. Scalability is an abstract attribute of software that refers to its ability to handle increased data throughput without modification. WebSphere handles scalability by allowing execution on a variety of hardware platforms that allow increased performance and clustering.

Servlets. Servlets are Java classes that run on Web servers to provide dynamic HTML content to clients. They take as input the HTTP request from the client and output dynamically generated HTML. For more information on servlets see <http://www.software.ibm.com/ebusiness/pm.html#Servlets>.

SOCKS. A SOCKS server is a proxy server that uses a special protocol, sockets, to forward requests. Transcoding Publisher connects through a SOCKS server that is configured with a firewall to manage network traffic and to protect your network from outside intrusion (it supports Versions 4 and 5 SOCKS servers).

SSL. Secure Sockets Layer. A secure protocol used for authentication and encryption. SSL can be used over HTTP, RMI, Telnet and other protocols.

Stand-alone Network Proxy. User uses the IBM WebSphere Transcoding Publisher as a normal proxy in his browser and the data that flows from the original source will be transcoded in the proxy according to the device and network profile needed.

Stylesheet Transcoder. Stylesheet Transcoder is the transcoder that selects the stylesheet and applies it to an input Extensible Markup Language (XML) document to produce a version that is appropriate for the target device.

Text Transcoder. Text Transcoder is the transcoder that can modify elements of a text document based on device, network, and potentially user preference information. The primary use of this Text Transcoder is to modify Hypertext Markup Language (HTML) documents to remove unsupported elements, reduce space usage, replace features such as images or frames with links, and otherwise tailor documents

to make them render more gracefully on constrained devices.

Transcoder. Transcoder is a program that modifies the content of a document.

Transcoding. Transcoding is a new technology that gives you the ability to make Web based information available on handheld and other new type devices economically and efficiently or on the slow network connections like a dial up modem connection. With transcoding, users receive information (text and images) tailored to the capabilities of the devices they are using and also tailored according to the capacity of the network being used.

It is also the process where the MEGs related to modifying the request, generating the original resource and all of the document (or resource) editing (or transcoding) occurs.

WAP. Wireless Application Protocol.

WAS. IBM WebSphere Application Server.

Web Application Servers. A Web application server is a software program designed to manage applications at the second-tier of three-tier computing, that is, the business logic components. A Web application server manages applications that use data from back-end systems, such as databases and transaction systems, and provides output to a Web browser on a client. For more information see <http://www.software.ibm.com/ebusiness/appsvsw.html>

WinGate. It is a proxy server. It is used in this redbook as a cache server in sample scenarios.

You can download an evaluate copy of WinGate from the Web on <http://www.wingate.com>

WTE. Web Traffic Express. An IBM caching proxy.

WTP. WebSphere Transcoding Publisher.

XML. XML, or eXtensible Markup Language, is a platform-independent and application-independent way of describing data using tags. XML (a subset of SGML) is similar to HTML in that it uses tags to describe document elements but different in that the tags describe the structure of the data rather than how the data

is to be presented to a client. XML has the facility to allow data providers to define new tags as needed to better describe the data domain being represented. For more information see <http://www.software.ibm.com/xml>.

XSL Stylesheets. XSL stylesheets are documents that describe a mapping between XML documents and visual data that can be presented to a client in a browser. XSL was a draft standard when this book was being written. The draft can be found at <http://www.w3.org/TR/WD-xs>.

Index

A

Administration Console 7, 17, 22, 24, 33, 59, 101
Advanced Function Presentation 14
AFP 14
alias 100, 110, 140
Aliases option 100
API 18, 19, 64

B

bean 127
browser 15, 19, 130, 136, 146

C

cache 5, 41, 51, 53, 54, 55, 59, 73
caching of documents 79
Criteria matching HTTP header 106
Criteria matching preferences 106

D

device 1, 2, 3, 4, 5, 8
Dial Network 3
DNS 77, 84, 91
DTD 25

E

Extensible Markup Language 4, 14, 25, 29
Extensible Stylesheet Language 14

F

filter 1, 6, 9, 13, 37, 69, 93, 94, 96, 109, 110, 113
firewall 5, 6, 39, 41, 51, 53, 54, 55, 57
framework 2, 7, 8, 14, 15, 19, 23

G

GIF 4, 16, 19, 20, 27, 39, 65, 107, 158, 203
Graphical User Interface 165

H

HDML 26
Host Publisher 9, 11, 314
Host Publisher Server 236
Host Publisher Studio 236
HTML 4, 7, 12, 65, 107, 123, 136, 138, 139, 144,

149

HtmlHandlerBean 123
HtmlReducerBean 123, 139
HTTP 5, 6, 41, 57, 64, 65, 73, 74, 75, 90, 129
HTTP reply 65
Hypertext Markup Language 4

I

IBM HTTP Server 96
IBM WebSphere Application Server 93
IBM WebSphere Transcoding Publisher 1, 93, 95,
96, 98, 109, 113
image 111, 158, 159, 161, 165, 203

J

JavaBeans 7, 9, 28, 99, 121, 122, 125, 126, 127,
132, 157, 163, 166, 198
JavaScript 16
JavaServer Pages 93, 95, 99, 121, 128, 142
JDBC compliant databases 236
JDK 45
JPEG 4, 16, 20, 27, 39, 65, 158
JSP 7, 28, 42, 95, 98, 99, 100, 111

L

Linux 2, 3, 7, 41, 132, 134
logging 9, 15, 35

M

MEG 18, 19, 20, 35, 36, 74, 81, 91, 123, 126, 166
MEGlets 19, 94, 166
Microsoft Explorer 3
MIME 19, 134, 139, 165, 166
MIME-filter 96, 97
MIME-type 93, 99, 110
multiple platforms 237

N

Netscape 3, 111
network profiles 3
Nokia 23, 59

O

Optional selection criteria 105

P

pageCompile 100
Palm device 148
Palm Pilot 3, 32, 158, 161
plug-ins 14, 16, 19, 34, 36, 98
Preference Aggregator 65
preference profiles 51
profile 2, 3, 5, 7, 9
proxy 9, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 84, 91, 97, 121

R

Request Viewer 9, 17, 70, 71, 88, 166, 167
Required selection criteria 105

S

Scalable Vector Graphics 14
Secure Sockets Layer 93
server 51, 52, 63, 64, 65, 67, 68, 69, 73
Server Setup 50, 51, 96
Server Setup wizard 97
servlet 19, 23, 28, 34, 42, 45, 53, 93, 94, 98, 99, 100, 108, 112, 113
Session pools 236
Snoop 38
SOCKS 54, 55, 57
SSL 68, 93
stylesheet 4, 7, 14, 16, 20, 22, 23, 25, 26, 29, 102, 123, 133, 163, 164, 167
Stylesheet Selection Criteria 104
SVG 14

T

Toolkit 8, 11, 17, 48, 59, 70, 157, 205, 314
tracing 7, 9, 35
transcoders 3, 7, 9, 14, 16, 17, 19, 20, 21, 22, 23, 27, 28, 42, 133
transcoding 1, 5, 6
Transcoding Publisher 2, 6, 73, 74, 75, 76, 77, 78
TranscodingFilter 94, 98, 99, 110
TranscodingPreamble 94, 98, 100, 110
Transform Tool 9, 128, 157

U

URL 19, 20, 30, 65, 91, 136, 146
User_Agent field 23
user-agent 37, 134, 139, 193, 202

User-Agent keyword 21, 165

W

W3C 14
WAP 3, 4, 11, 12, 16, 23, 25, 29, 65, 161, 173, 314
WAP gateway 65
WAS 93, 94, 95, 96, 97, 98, 99, 109, 110
WBI 16, 18, 19, 36, 123, 126, 127
Web 1, 4, 5, 6, 41, 52, 110, 113
Web server 96, 98
WebSphere 42, 43, 45, 46, 52, 53, 60
WebSphere administration 110
WebSphere Application Server 94, 96, 97, 98
WebSphere Transcoding Publisher 3, 9
Windows CE 3, 32
wireless 21
Wireless Application Protocol 25, 29
Wireless Markup Language 4, 16, 25, 29, 65, 123
Wireless Network 3
Wireless Phones 3
WML 4, 7, 42, 65, 123, 161, 163
WorkPad 21
WTE 73, 81, 83, 89
WTP 30, 66, 69, 72, 93, 129, 165, 193, 201, 287

X

XML 4, 7, 65, 102, 112, 113, 123, 133, 153, 163, 287, 288
XML-Capable 3
XmlHandlerBean 123, 126
XSL 14, 20, 25, 30

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at ibm.com/redbooks
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number	SG24-5965-00
Redbook Title	IBM WebSphere Transcoding Publisher V1.1 Extending Web Applications to the Pervasive World
Review	<div></div> <div></div> <div></div> <div></div> <div></div> <div></div>
What other subjects would you like to see IBM Redbooks address?	<div></div> <div></div> <div></div>
Please rate your overall satisfaction:	<input type="radio"/> Very Good <input type="radio"/> Good <input type="radio"/> Average <input type="radio"/> Poor
Please identify yourself as belonging to one of the following groups:	<input type="radio"/> Customer <input type="radio"/> Business Partner <input type="radio"/> Solution Developer <input type="radio"/> IBM, Lotus or Tivoli Employee <input type="radio"/> None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	<input type="radio"/> Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. ibm.com/privacy/yourprivacy/



IBM WebSphere Transcoding Publisher V1.1 : Extending Web Applications to the Pervasive World

(0.5" spine)

0.475" <-> 0.875"

250 <-> 459 pages



IBM WebSphere Transcoding Publisher V1.1

Extending Web Applications to the Pervasive World

**Enable universal
data access to
improve business
communications**

**Write once - reach
anyone, anytime,
anywhere**

**Quickly support your
pervasive devices**

This redbook helps you to understand the IBM WebSphere Transcoding Publisher (WTP) product. It focuses on architectures and technologies implemented in this product and helps you plan, install and configure WTP in the supported models. For example, you will find information on how to configure WTP to run as a proxy server, as a WebSphere Application Server (WAS) filter, or simply as JavaBeans within user applications.

You will also find numerous configuration scenarios showing ways to set up the IBM WebSphere Transcoding Publisher Version 1.1 to adapt and reformat your application content (HTML, XML and images) so it can be accessed from devices such as WAP phones supporting WML, Palm Pilots, Windows CE and WorkPads.

A basic knowledge of Java technologies such as servlets, JavaBeans, Java Server Pages (JSPs) and the terminology used in Web publishing is assumed.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks